

# Softwaretechnik

Sommersemester 2024

Thomas Slotos, M.Sc.

Die Präsentation gibt nur ein Teil der Inhalte der Vorlesung wieder und dient als Leitfaden, nicht aber der vollständigen Dokumentation. Wesentliche Konzepte werden im Rahmen der Vorlesung entwickelt und erläutert.

- CV
  - Lehrkraft für besondere Aufgaben
  - Mitglied im IIS
  - Studium der Wirtschaftsinformatik
  - Studium des Software Engineering (England)
  - Mitarbeit an einem Forschungsprojekt im Bereich Softwaretechnik (Hamburg)
- Lehr- und Forschungsgebiete
  - Softwaretechnik
  - Verteilte Anwendungssysteme
  - Web- und mobile Anwendungsentwicklung
  - Komponentenorientierte Softwarearchitektur
- Kontakt
  - [thomas.slotos@leuphana.de](mailto:thomas.slotos@leuphana.de)
  - Universitätsallee 1, Gebäude 4, Raum 303



## Agenda

- Definition Softwaretechnik
- Allgemeine Systemtheorie
- Projektaufgabe
- Anforderungsanalyse
  - Anforderungen
  - Lasten-/Pflichtenheft
  - Analysemuster
- Unified Modeling Language (UML)
  - Verhaltensdiagramme
    - Use Case Diagramm
    - Aktivitätsdiagramm
    - Zustandsdiagramm
  - Strukturdiagramme
    - Klassendiagramm
    - Paketdiagramm
- CASE-Tool

**"Sagst du's mir, so vergesse ich es,  
zeigst du's mir, so merke ich es mir,  
läßt du mich teilnehmen,  
so verstehe ich es!"**

Chinesisches Sprichwort

## **Definition Softwaretechnik**

## Software

„Computer programs, procedures, and possible associated documentation and data pertaining to the operation of a computer system.“

IEEE Standard 610.12 (1990)

## Software-System

„Ein System, dessen Systemkomponenten und Systemelemente aus Software bestehen“

H. Balzert, 2001

## Besonderheiten von Software

- Immateriell
- Kopie und Original sind identisch
- Kein Verschleiß
- Schwer zu „vermessen“
- Wird nicht durch physikalische Gesetze begrenzt

(siehe H.Balzert, 2001)

## Definition (Software Engineering)

„The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them“

Berry Boehm, 1976

„The systematic approach to the development, operation, maintenance, and retirement of software“

IEEE Standard Glossary of Software Engineering Terminology, ANSI 1983

## Definition (Software Engineering)

„Zusammenfassend kann man Software-Engineering als die Wissenschaft der systematischen Entwicklung von Software, beginnend bei den Anforderungen bis zur Abnahme des fertigen Produkts und der anschließenden Wartungsphase definieren.

Es werden etablierte Lösungsansätze für Teilaufgaben vorgeschlagen, die häufig kombiniert mit neuen Technologien, vor Ihrer Umsetzung auf ihre Anwendbarkeit geprüft werden. Das zentrale Mittel zur Dokumentation von Software-Engineering-Ergebnissen sind UML-Diagramme.“

S. Kleuker, 2013

## Definition (Softwaretechnik)

„Softwaretechnik, Softwaretechnologie, Software-Engineering das, – Teilgebiet der Informatik, das sich mit Methoden und Werkzeugen für das ingenieurmäßige Entwerfen, Herstellen und Implementieren von Software befasst“

Brockhaus 2008

„Fachgebiet der Informatik, das sich mit der Bereitstellung und systematischen Verwendung von Methoden und Werkzeugen für die Herstellung und Anwendung von Software beschäftigt.“

Hesse et. al., 1984 (nach Balzert 2009)

„Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen. Zielorientiert bedeutet die Berücksichtigung z.B. von Kosten, Zeit, Qualität“

Balzert, 2009

„Der, die, das,  
**wer?, wie?, was?,**  
**wieso?, weshalb?, warum?...**  
Wer nicht fragt bleibt dumm!

Tausend tolle Sachen,  
die gibt es überall zu seh'n,  
manchmal muss man fragen,  
um sie zu versteh'n!“



Quelle: Text zur Titelmelodie der deutschen Sesamstrasse ab 1973

## W-Fragen zur Strukturierung eines Themenbereichs

- Warum? → Frage nach dem Grund / Ursache (vergangenheitsbezogen)
- Wozu? → Frage nach dem Ziel / Zweck (zukunftsbezogen)
- Wer? → Frage nach den Personen / Rollen
- Womit? → Frage nach den Werkzeugen / Mitteln
- Was? → Frage nach den Artefakten (Ressourcen bzw. Teilergebnissen)
- Wie? → Frage nach der Vorgehensweise
- Wann? → Frage nach dem zeitlichen Einsatz / Dauer
- Wo? → Frage nach den (Einsatz)ort oder Raum

## W-Fragen zur Strukturierung eines Themenbereichs

**Warum?** Der zu analysierende Teilausschnitt der realen Welt ist komplex. Viele Elemente und deren Verknüpfungen untereinander (siehe „Allgemeine Systemtheorie“) sorgen dafür, dass die zu untersuchenden Strukturen sich in der Zeit ändern.

**Wozu?** Gewonnene Erkenntnisse sollen es ermöglichen komplexe (Software-)Systeme in hoher Qualität bei geringen Kosten zu bauen. Ein wesentliches Ziel ist es ein ingenieurmäßiges Vorgehen anzubieten um ein zukünftiges Softwareprojekt aufgrund einer strukturierten und geplanten Form erfolgreich abzuschließen.

**Wer?** Diverse Rollen auf Seiten des Kunden, aber auch im Softwareentwicklungsteam müssen miteinander kommunizieren um Anforderungen umzusetzen. Diejenigen, die aus unterschiedlichen fachlichen Interessensbereichen Anforderungen definieren werden Stakeholder genannt.

**Womit?** Zur Verarbeitung von Ressourcen und Erarbeitung von (Teil-)ergebnissen werden Werkzeuge wie z.B. CASE-Tool eingesetzt um schrittweise die Anforderungen von der Sprache der Stakeholder in die Sprache der Informatik (z.B. UML) umzusetzen.

## W-Fragen zur Strukturierung eines Themenbereichs

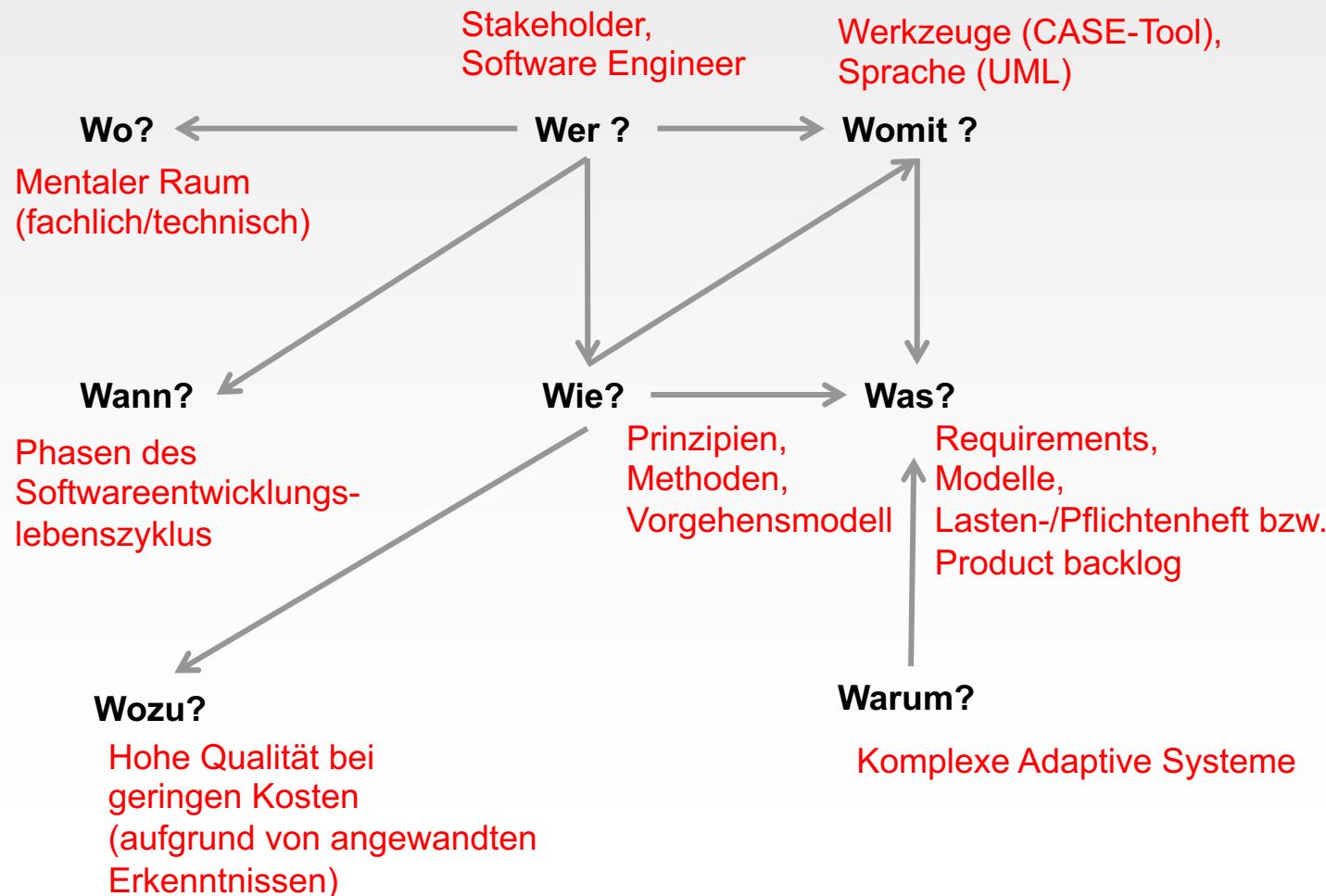
**Was?** Aufgrund der Komplexität – auch des zu erarbeitenden Softwareproduktes – werden auf Basis von Anforderungen aus unterschiedlichen Sichten Modelle der zukünftigen Software erstellt und diese in Dokumenten (z.B. Lasten-/Pflichtenheft) beschrieben. Diese Teilergebnisse heißen Artefakte.

**Wie?** Eine genau vorgegebene Vorgehensweise (Methode) soll es ermöglichen ökonomisch hoch qualitative Software zu erstellen. Die Vorgehensweise beschreibt wie die allgemeinen Anforderungen der Stakeholder in ein konkretes Programm umgesetzt werden können.

**Wann?** Als Querschnittsaufgabe begleitet die Softwaretechnik das Softwareentwicklungsprojekt von der Analyse über den Entwurf und die Realisierung bis zum Einsatz/Wartung der Software. D.h. Softwaretechnik ist allgegenwärtig, wenn auch in unterschiedlicher Intensität.

**Wo?** Zumindest in der ersten Phase (Systemanalyse) hat der Software Engineer aufgrund der Anforderungserhebung intensiv mit den Stakeholdern (auch vor Ort) zu tun. Je nach Vorgehensmodell der Softwareentwicklung bleibt es bei einem intensiven Kontakt mit dem Kunden vor Ort oder es kommt zur verringerten Kundenkontakt.

## Essentielles Begriffsmodell

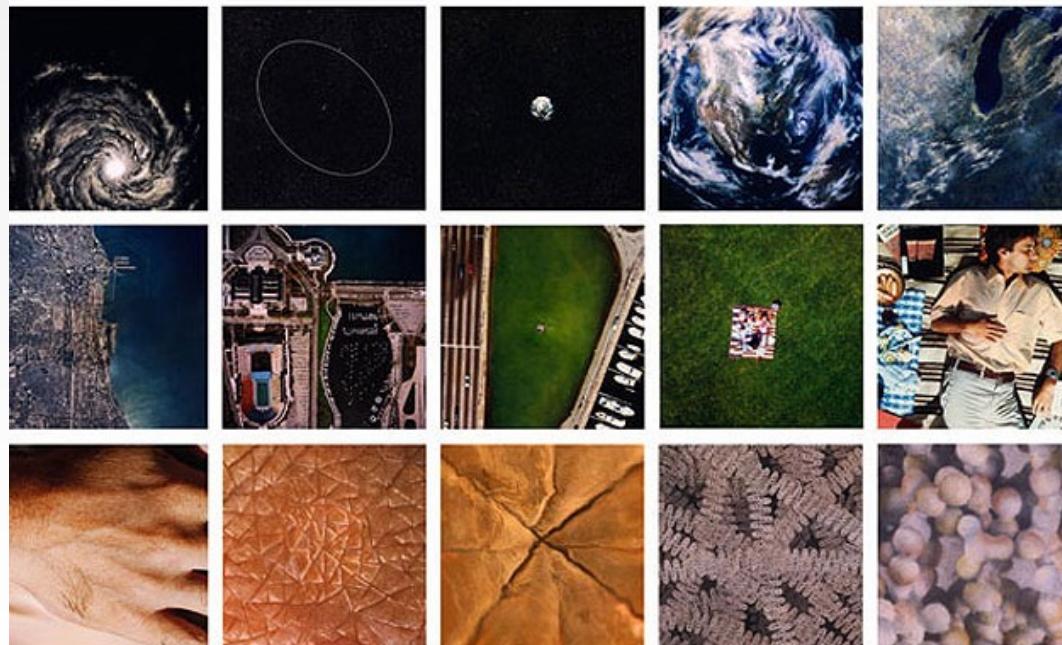


# Warum?

# Allgemeine Systemtheorie

“The Universe is a hierarchy of systems; that is, simple systems are synthesized into more complex systems from subatomic particles to civilizations.”

Skyttner, L., General Systems Theory, 2001



<http://www.iconeye.com/images/2014/04/Icon120-Powersof10-inside.jpg>; Zugriff: 20.04.2015

Quelle: Eames, C. Eames, R.; The Power of Ten, 1977

“The concept of a system is one of the most widely used concepts in science, particularly in recent times. It is encountered in nearly all the fundamental fields of science, such as physics, chemistry, mathematics, logic, cybernetics, economics, linguistics, biology, psychology, as well as in the majority of engineering branches.”

Klir, G.; The general system as a methodological tool, General Systems, Vol. 10, 1965, p. 29

Aus wissenschaftlicher Sicht wird die „Welt“ als **aufeinander aufbauende und miteinander interagierende Systeme** interpretiert. In der „**Allgemeinen Systemtheorie** (von Bertalanffy (1950))“, die transdisziplinär aufgestellt ist und daher den Anspruch einer „Science of Science“ hat, werden die allgemeinen Kriterien von Systemen und deren Interaktionen untersucht.

Da die „Welt“ durch **raum-zeitliche Dimensionen** gekennzeichnet ist, kommen diese Eigenschaften in Systemen (und damit mit auch in Modellen) in Form von (räumlichen) **Strukturen** und (zeitlichem) **Verhalten** vor.

Untersucht man Systeme in diesen beiden Dimensionen, dann stellt man fest, dass sie **komplex** strukturiert sind und sich in zeitlicher Hinsicht **evolutionär** verändert haben.

Systeme benötigen für ihre „Lebensfähigkeit“ **Materie, Energie und Information**. Dabei stellt Materie die physisch vorhandenen Elemente dar, die durch Energie umgeformt werden können. Information dient als „Anleitung“ wie diese Transformation zu vollziehen ist.

Die Definition, was ein System ausmacht ist **subjektiv**, d.h. von der **Rolle** abhängig in dem sich der Mensch bewusst oder unbewusst befindet.

Systeme sind entweder **gegenständlicher** oder **gedanklicher** Natur. Gegenständliche Systeme sind entweder **natürliche** oder **künstliche** (vom Menschen geschaffene) Systeme.

In der **Wirtschaftsinformatik** werden sowohl Systeme aus dem Anwendungsbereich „Wirtschaft“ als auch „Informatik“ untersucht.

Im Wirtschaftsbereich stellen die funktionalen Organisationseinheiten wie z.B. Einkauf, Produktion, Vertrieb, Controlling, Logistik zu untersuchende Systeme dar.

Diese Teilsysteme aus dem **Wirtschaftsbereich** sind in Form von IT-Systemen realisiert. Diese können z.B. ein Customer Relationship Management (CRM) oder ein umfassenderes wie z.B. ein Enterprise Resource Planning (ERP) System sein.

Innerhalb der **Informatik** können Systeme ebenfalls nach Funktionen unterschieden werden. Es gibt ein Datenbankmanagementsystem (DBMS), ein Netzwerksystem, ein Graphical User Interface (GUI) System.

Siehe auch:

- Ropohl, G., Allgemeine Systemtheorie, Edition Sigma, 2012
- Ferstl, O., Sinz, E.: Grundlagen der Wirtschaftsinformatik, Kapitel 2.1, Oldenbourg, 2015
- Laudon, K., Laudon, J., Schoder, D.; Wirtschaftsinformatik – Eine Einführung, Kapitel 2.4, 2. Aufl., Pearson, 2015
- Trier, M. et. al.; Systemtheorie und Modell; in: Krallmann, H. et. al.; Systemanalyse im Unternehmen – Prozessorientierte Methoden der Wirtschaftsinformatik; 2013

„... bezeichnet allgemein eine Gesamtheit von Elementen, die so aufeinander bezogen bzw. miteinander verbunden sind und in einer Weise wechselwirken, dass sie als eine aufgaben-, sinn- oder zweckgebundene Einheit angesehen werden können.“

Quelle: <http://de.wikipedia.org/wiki/System>; Zugriff: 10.09.2015

## Generelles

- Umwelt (andere Systeme)
- Grenze (Innen/Außen)
- Schnittstellen
- Eingangs-/Ausgangsgrößen
- zweckgebundene Einheit (durch Struktur und Verhalten)
- Zustand

## Strukturelle Eigenschaften

- Elemente
- Beziehungen
- Organisation (Anordnung Elemente und Beziehungen)

## Verhaltenseigenschaften

- Ereignisse
- Aktionen
- Kontrollstrukturen

# Warum? - Systembegriff (2)

## Generelles

Ein System bildet meist eine **zweckgebundene Einheit**, die durch seine innere Struktur und Verhalten realisiert wird. Struktur und Verhalten stellen eine **komplementären Beziehung (Dichotomie)** dar, d.h. keine Struktur ohne Verhalten und umgekehrt. Dabei ist es so, dass durch Verhaltensänderung des Systems seine innere Struktur geändert wird. Betrachtet man die Struktur und das Verhalten des Systems zu einem Zeitpunkt, so nennt man dies den **Zustand** des Systems.

Sowohl die innere Struktur als auch die inneren Prozesse, die das Verhalten des Systems darstellen, sind durch eine physische oder logische **Grenze** von seiner Umwelt abgegrenzt.

**Umwelt** stellt alles dar was außerhalb des Systems ist. Umwelt liegt in Form von anderen Systemen vor mit denen das System in Wechselwirkung steht.

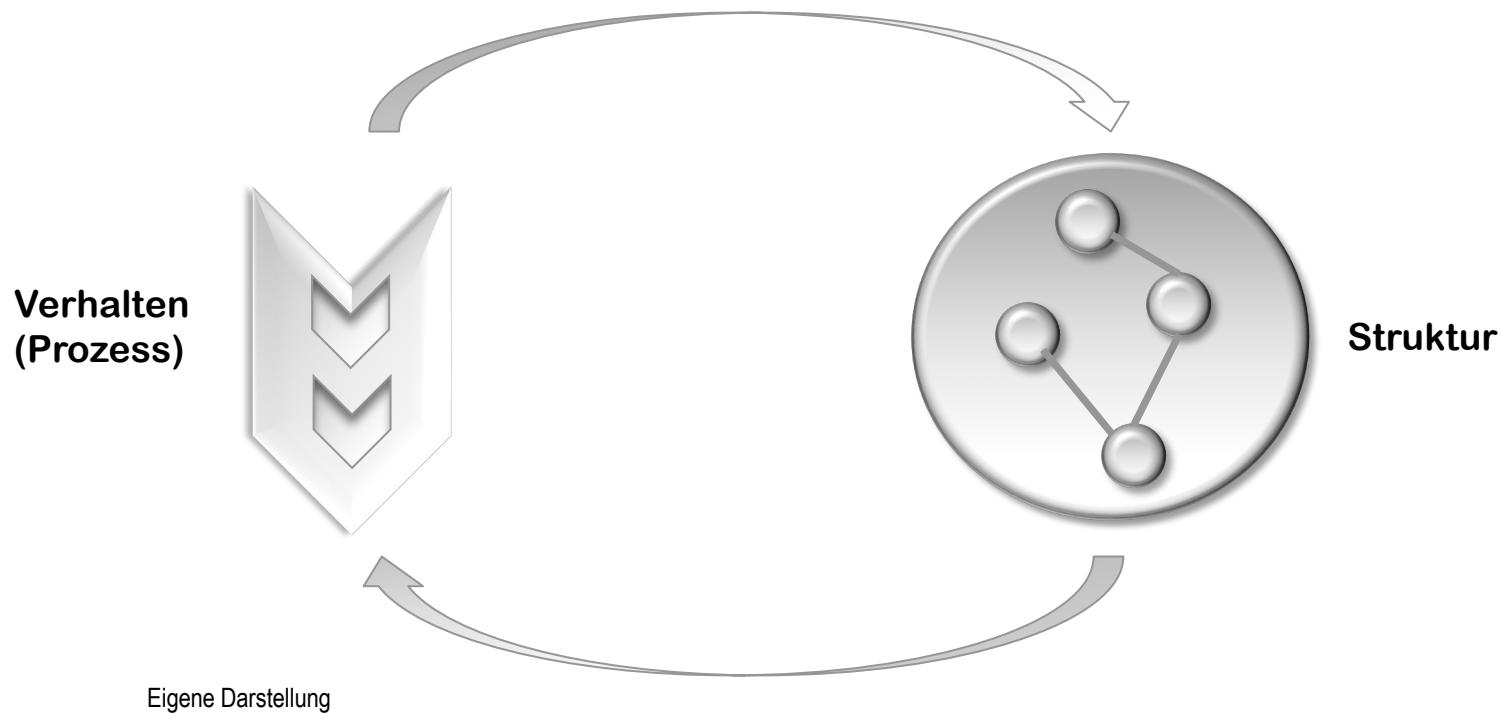
Dynamische Systeme, also Systeme, die sich in der Zeit verändern, benötigen zur Aufrechterhaltung ihrer Funktionalität **Eingangsgrößen** in Form von Materie, Energie, Information.

Das System selbst produziert **Ausgangsgrößen** in der gleichen Form. Nicht alle diese Formen müssen als Eingangs- und Ausgangsgrößen vorliegen. Das System bezieht diese Eingangs- und Ausgangsgrößen aus seiner Umwelt.

Den Ort an dem die Eingang- und Ausgangsgrößen ausgetauscht werden, nennt man **Schnittstelle**. Nur definierte Größen, die für die Aufrechterhaltung des Systems wichtig sind gelangen über die Schnittstelle in das System. Welche Größen dies sind, bestimmt das System und nicht die Umwelt. Bei den Ausgangsgrößen verhält es sich identisch.

Damit steht das System und die Umwelt in einer **komplementären Beziehung**, d.h. kein System ohne Umwelt und umgekehrt.

## Struktur und Verhalten



„.... structure is an arrangement in space; and process is an arrangement in time..“

Ing, D., Rethinking Systems Thinking: Learning and Coevolving with the World, 2013

„The systems paradigm advanced a framework that focused on commonalities in structure and process..“

Ruben, B., Foreword in: Ackoff, R., Emery, F.; On purposeful systems, 4th ed., 2009

## Struktur und Verhalten

"We use the concept of process to characterize qualities of a phenomenon which we perceive as related to change; change in time and space, and in the sense development and transformation.

The concept of structure is used to characterize qualities of a phenomenon which we perceive as fixed and stable.

We focus on the temporary stability, at the same time maintaining that the perceived qualities are changeable."

Mathiassen, L.; Systems, Processes and Structures - a Contribution to the Theoretical Foundation of Information Systems Development and Use, 1987

"... systems thinking is the art and science of making reliable inferences about *behavior* by developing an increasingly deep understanding of the underlying *structure*."

Richmond, B.; An Introduction to System Thinking with iTank, 2004

"Once we see the relationship between structure and behavior, we can begin to understand how systems work ..."

Meadows, D., Thinking in systems, 2008

## Strukturelle Eigenschaften

Jedes System besteht aus Elementen und Beziehungen zwischen den Elementen.

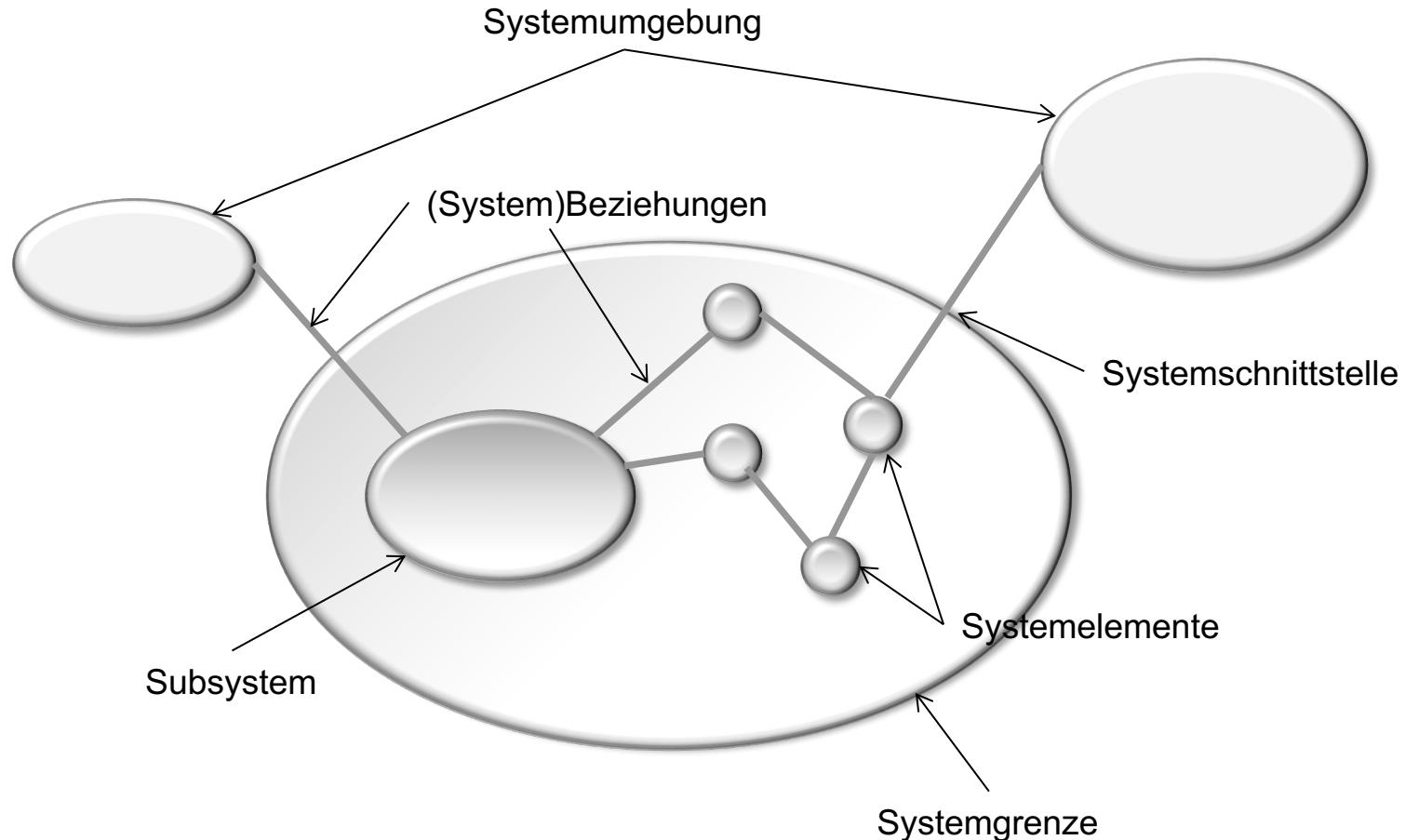
Was **Elemente** des Systems sind, hängt von der Betrachtung (Abstraktionsniveau) ab. Das Abstraktionsniveau ist von der Fragestellung/Aufgabe abhängig mit der ein Betrachter das System untersucht. Analysiert man Elemente wieder genauer so stellt sich heraus, dass auch Elemente wieder Systeme sind. D.h. der komplexe Aufbau von Systemen kommt durch die hierarchische Ineinanderschachtelung von Systemen zu Stande.

Elemente stellen keine lose Sammlung innerhalb des Systems dar. Die Funktion und damit das Verhalten des Systems in der Umwelt kommt nur dadurch zu Stande, dass die Elemente innerhalb des System miteinander in **Beziehung** stehen. Beziehung heißt, dass es zwischen den Elementen eine Verbindung gibt mittels derer Materie, Energie, Information ausgetauscht wird. Nicht in jedem System werden alle 3 Arten ausgetauscht.

Die Art und Weise dieser Anordnung von Elementen und Beziehungen nennt man (innere) **Organisation** des Systems.

In der **Informatik** werden die Strukturen eines Systems häufig in graphischer Form dargestellt. Hierzu gehören u.a. das Entity-Relationship-Diagramm für relationale Datenbanken und die Strukturdiagramme der UML wie Klassen-, Paket-, Deployment-Diagramm.

## System und Struktur



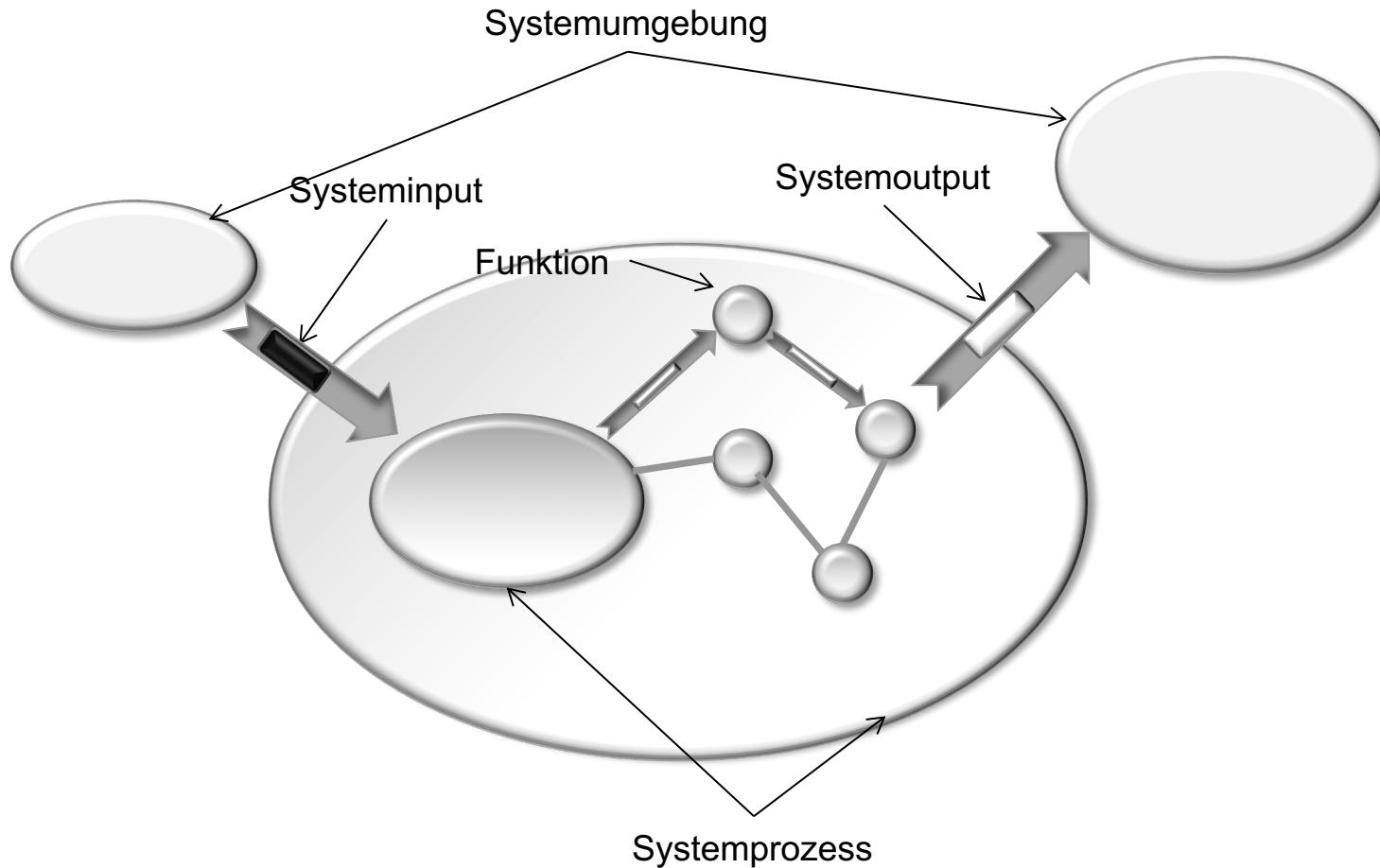
## Verhaltenseigenschaften

Die **Funktion** des Systems wird durch die in dem System ablaufenden Prozesse realisiert. Dabei stellt ein **Prozess** eine zeitliche Abfolge von Aktivitäten unter Benutzung der Elemente innerhalb des Systems dar. Jeder Prozess benötigt zu seinem Ablauf Eingangsgrößen, die über die **Schnittstellen** des Systems in das System gelangen. Produkte eines Prozesses werden wieder, soweit sie innerhalb des Systems nicht gebraucht werden, als Ausgangsgrößen an die Umwelt abgegeben.

Somit steht das System in **Wechselwirkung** mit der Umwelt.

In der **Informatik** wird das Verhalten eines Systems häufig in graphischer Form dargestellt. Hierzu gehören u.a. ein eEPK- bzw. ein BPMN-Diagramm. In der UML gehören hierzu die Verhaltensdiagramme wie Aktivitäts-, Zustands- und Sequenzdiagramm.

## System und Verhalten



## Systemarten

- **Reales System**

„Ein reales System ist dadurch gekennzeichnet, dass seine Komponenten real (materiell, energetisch) sind.“

Quelle: Ferstl, O., Sinz, E.: Grundlagen der Wirtschaftsinformatik, 7.Aufl., Oldenbourg, 2015

- **Informationssystem**

„An information system is a system that collects, stores, processes, and distributes information.“

Quelle: Antoni, O.; Conceptual Modeling of Information Systems, Springer, 2007

## Sichten

- **Außensicht (Blackbox)**

„In der Außensicht wird ein System als nicht mehr weiter detaillierbar betrachtet. Entsprechend ist auch nur das (äußere) Verhalten des Systems gegenüber seiner Umwelt sichtbar ... In der Außenansicht werden die Schnittstellen eines Systems beschrieben, über die es mit benachbarten Systemen in Beziehung stehen kann.“

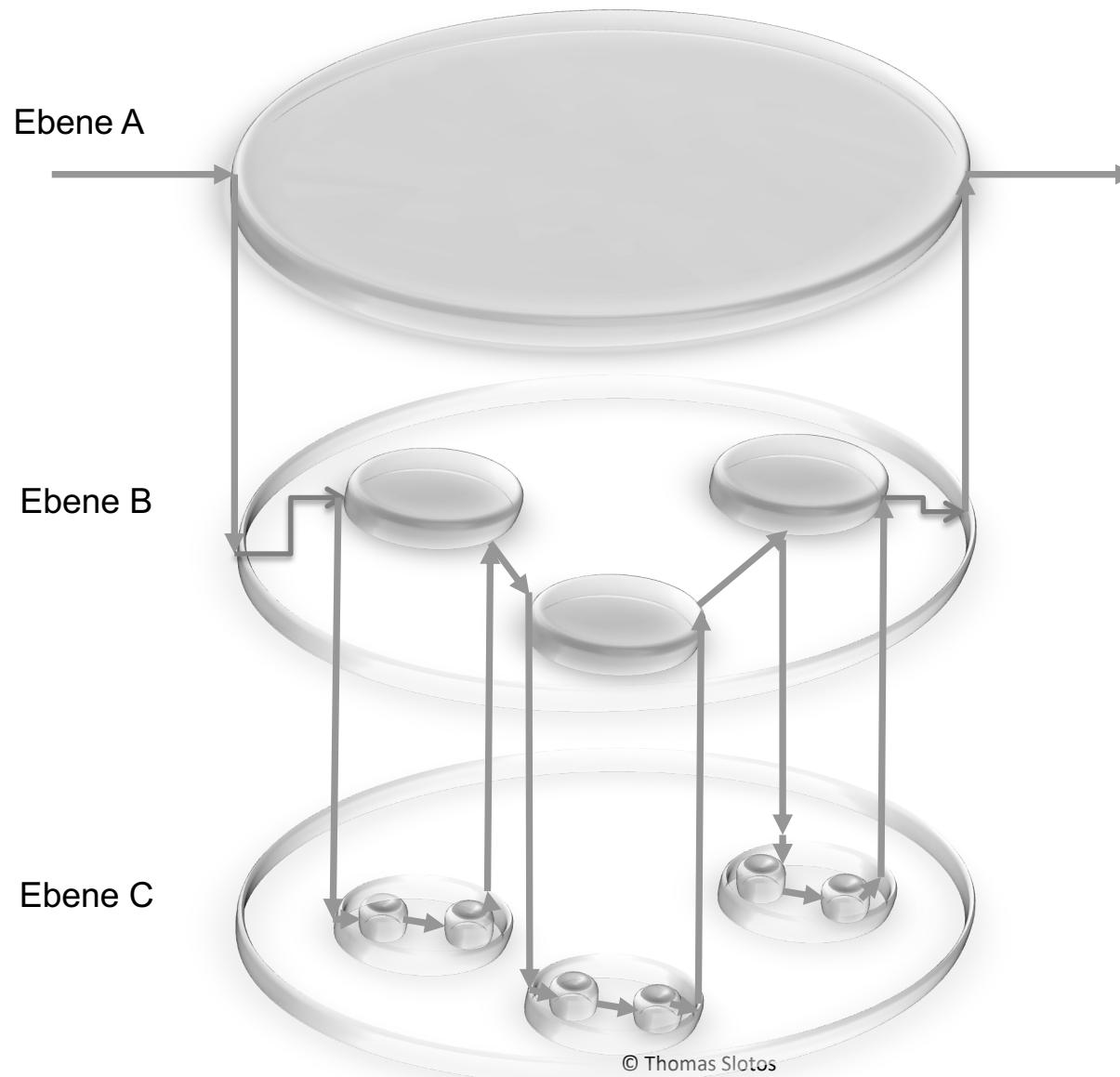
- **Innensicht (White box)**

„Die Innenansicht eines Systems betrachtet die (innere) Struktur und das Verhalten seiner Komponenten und Teilsysteme. ... Die Komponenten eines Systems werden entweder als elementar betrachtet oder als Teilsysteme, die wiederum aus ihrer Außensicht beschrieben werden. Die Beschreibung der Innensicht eines Teilsystems muss mit der zugehörigen Außenansicht verträglich sein, d.h. das äußere Verhalten realisieren.“

„Die mehrstufige, hierarchische Systembeschreibung bei gleichzeitiger Unterscheidung von Außen- und Innensicht ist eines der wichtigsten Hilfsmittel zur Bewältigung der Komplexität von Systemen.“

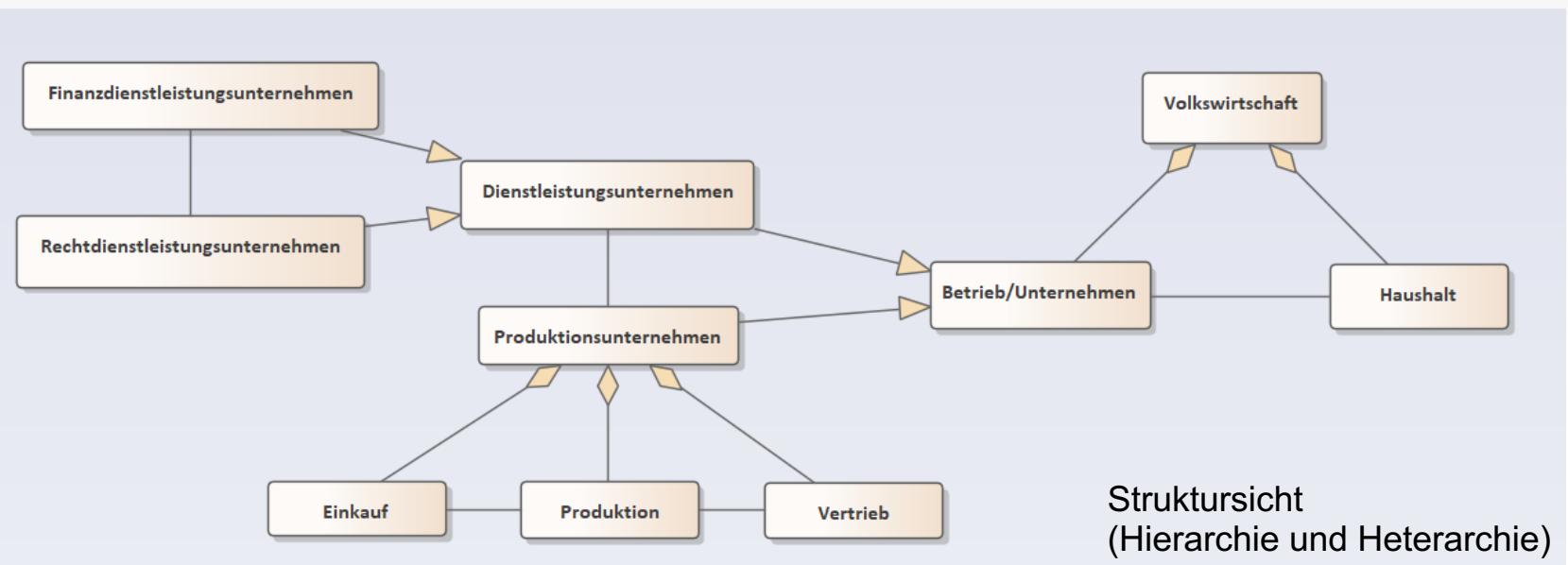
Quelle: Ferstl, O., Sinz, E.: Grundlagen der Wirtschaftsinformatik, 7.Aufl., Oldenbourg, 2015

## System – Blackbox/White box und Hierarchie/Heterarchie



## Systemrollen in einer Hierarchie / Heterarchie

Ein **betrachtetes System** ist in der Regel Teil eines größeren Systems. Das größere System stellt damit ein sogenanntes **Supersystem** dar. Das betrachtete System kann auch in Teilsysteme, sogenannte **Subsysteme**, aufgeteilt werden. Das betrachtete System und alle Super- und Subsysteme stellen eine Hierarchie dar. In einer Hierarchie mit seinen Ebenen, sind die Begriffe Super- bzw. Subsystem relativ in Bezug zum jeweils ausgewählten und damit betrachteten System. Auf jeder Ebene gibt es Beziehungen zwischen **benachbarten Systemen** (Heterarchie).

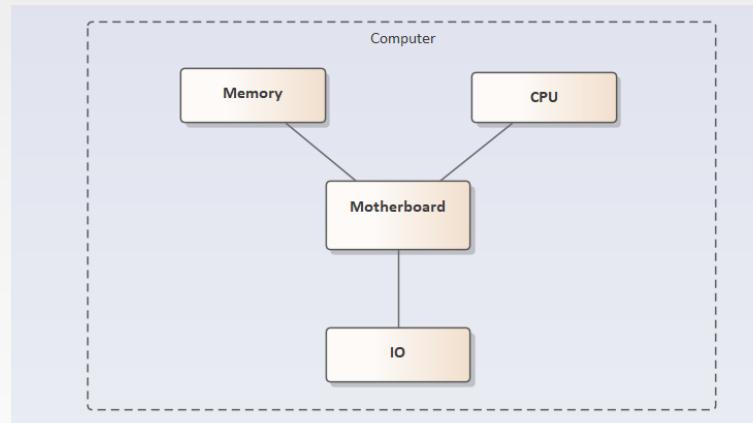


# Warum? - Systembegriff (13)

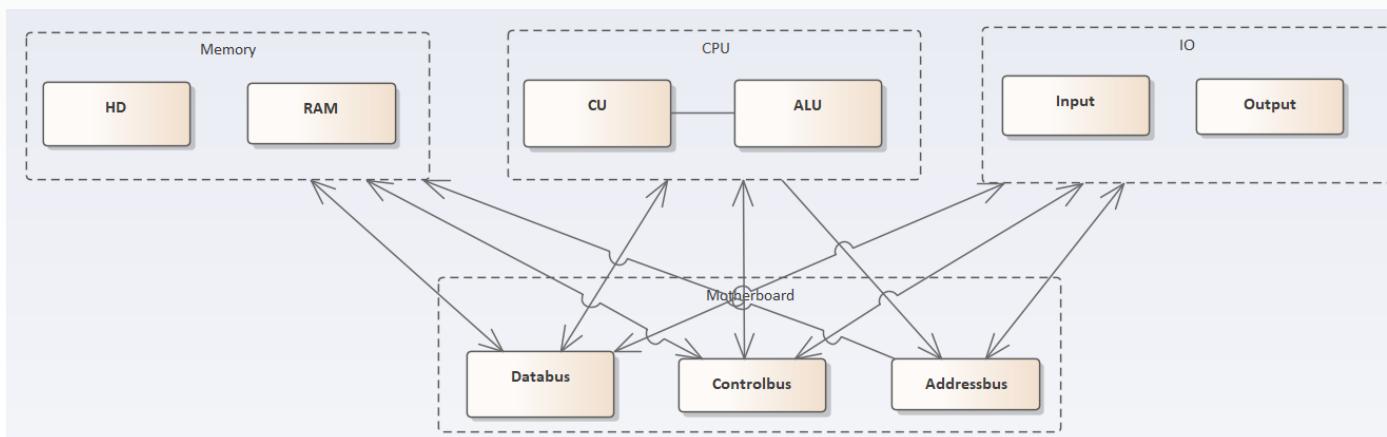
## Struktur



Ebene 1



Ebene 2

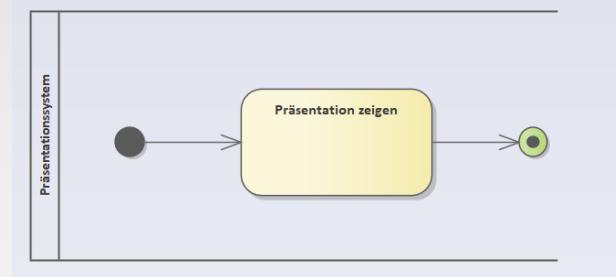


Ebene 3

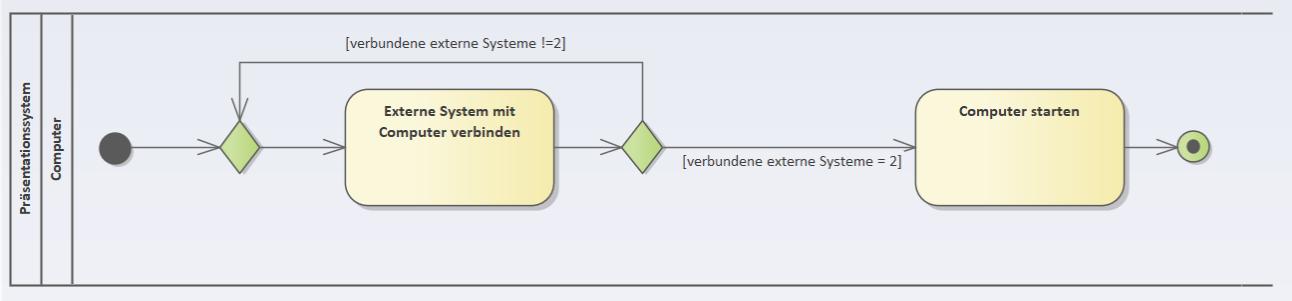
# Warum? - Systembegriff (14)

## Verhalten

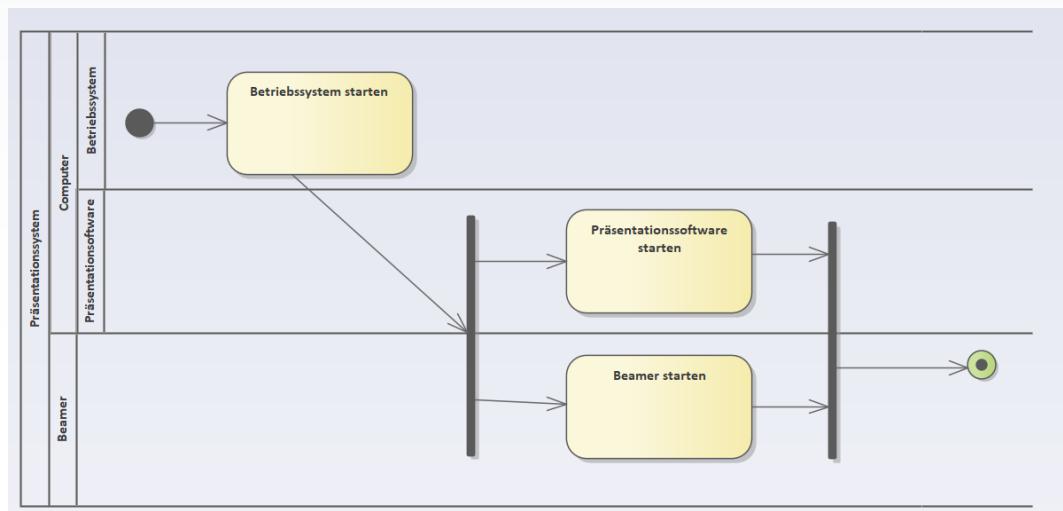
Ebene 1



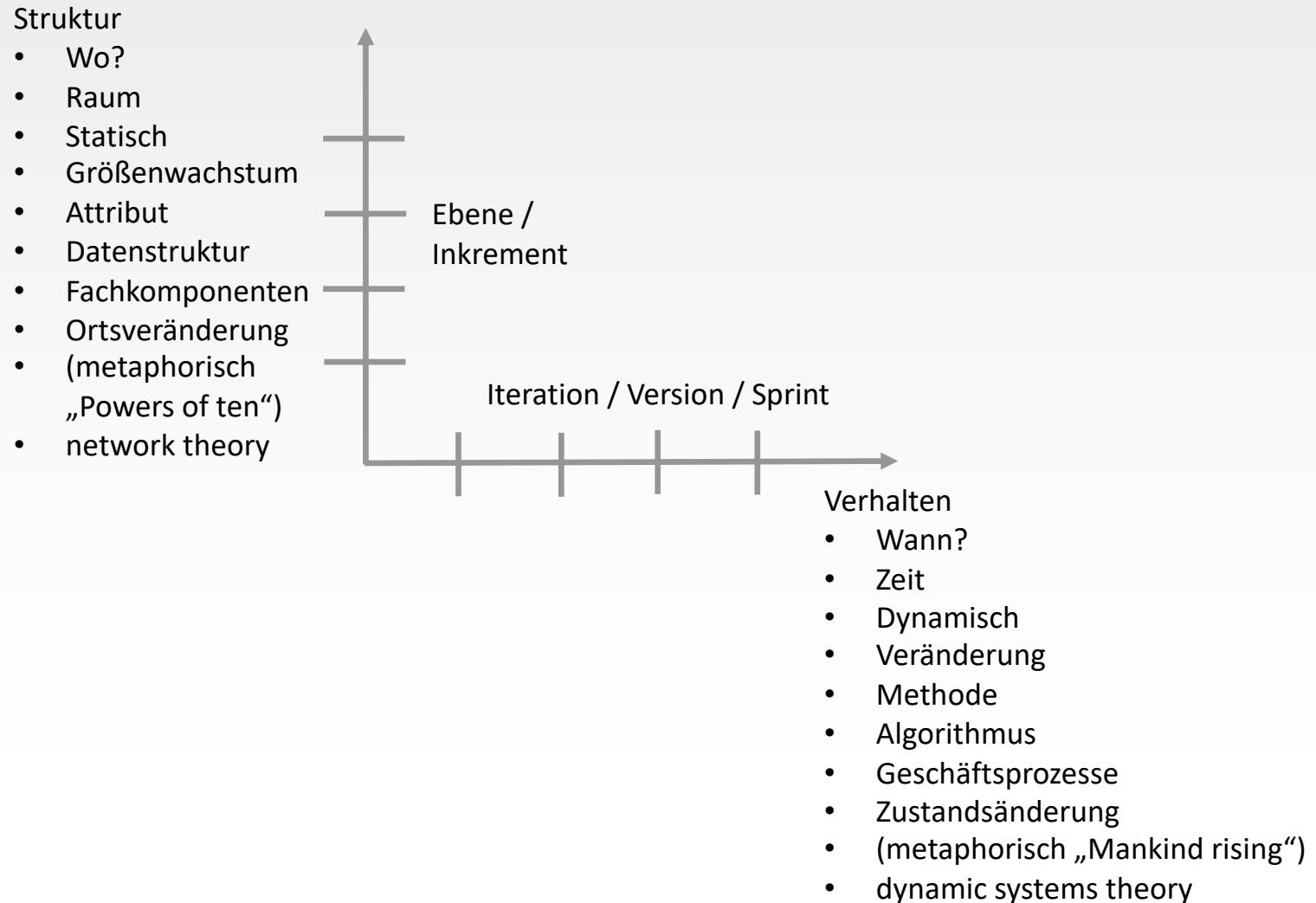
Ebene 2



Ebene 3



## Eigenschaften der Struktur- und Verhaltensdimension im Bereich Wirtschaftsinformatik



## Bestandteile

„The three basic entities that constitute the media on which systems operate are

1. *Information*: the content of all knowledge and communication,
2. *Material*: the substance of all physical objects, and
3. *Energy*: energizes the operation and movement of all active system components.“

Quelle: Kossiakoff, A. et al.; Systems Engineering principles and practice, 2nd ed., 2011

“Every system must find its place within its larger system, and the two systems must communicate via exchanges of information, material, or energy.”

Quelle Hatley, D. et. al. Process for system architecture and requirements engineering, 2000

“Information is considered as a fundamental building block of reality, along with matter and energy.”

Quelle: Meijer, D.; Information: what do you mean? On the formative element of our universe; 2013

## System und Modell

"It is generally agreed that "system" is a *model* of general nature, that is, a conceptual analog of certain universal traits of observed entities."

Quelle: Bertalanffy, L., The History and Status of General Systems Theory, 1972

[http://www.communicationcache.com/uploads/1/0/8/8/10887248/  
the\\_history\\_and\\_status\\_of\\_general\\_systems\\_theory.pdf](http://www.communicationcache.com/uploads/1/0/8/8/10887248/the_history_and_status_of_general_systems_theory.pdf)

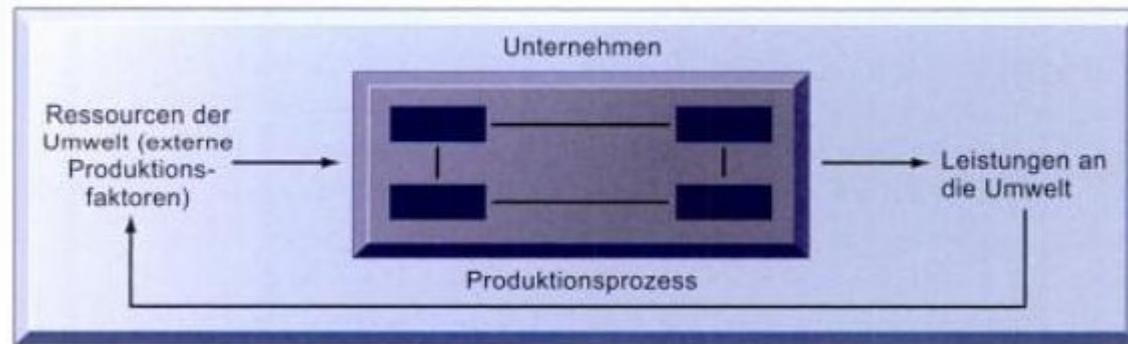
## Unternehmen

- **strukturorientierte Definition**

„Eine stabile, formale, soziale Struktur, die Ressourcen aus der Unternehmensumwelt und zur Erzeugung von Produkten verwendet.“

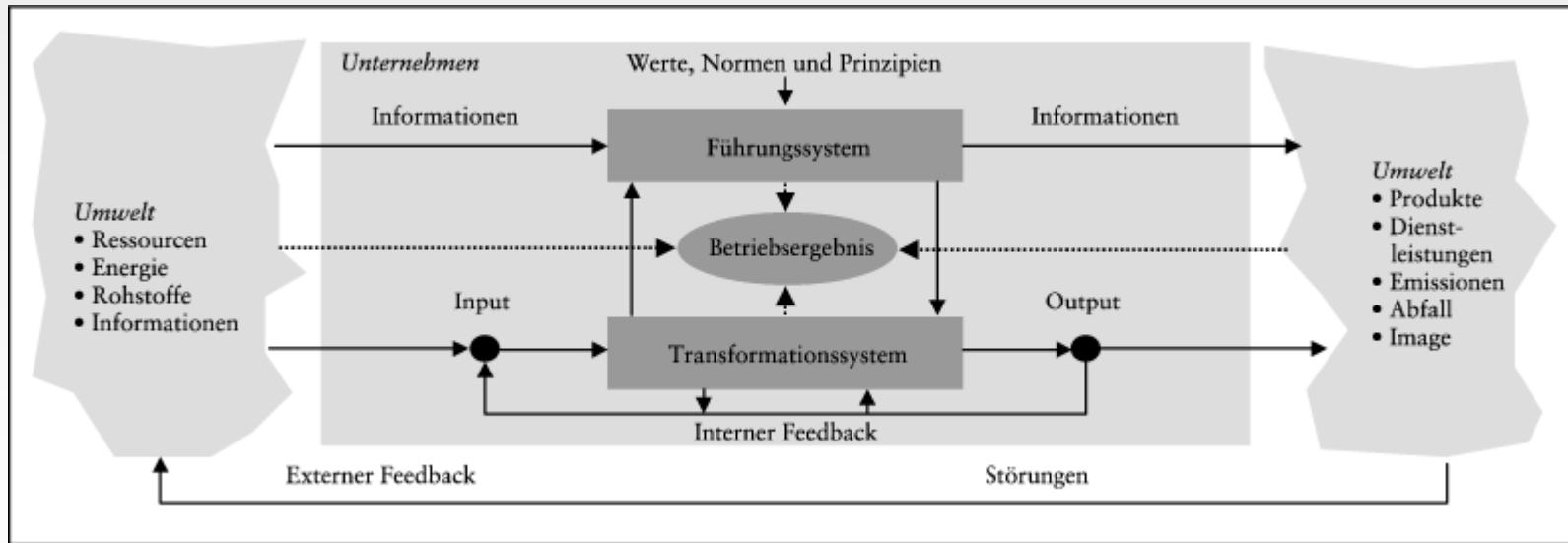
- **verhaltensorientierte Definition**

„Eine Sammlung von Rechten, Privilegien, Verpflichtungen und Verantwortlichkeiten, die im Laufe der Zeit durch Konflikte und Konfliktbewältigung ausgewogen verteilt wurden.“



Quelle: Laudon, K., Laudon, J., Schoder, D.; Wirtschaftsinformatik – Eine Einführung, 2. Aufl., Pearson, 2009

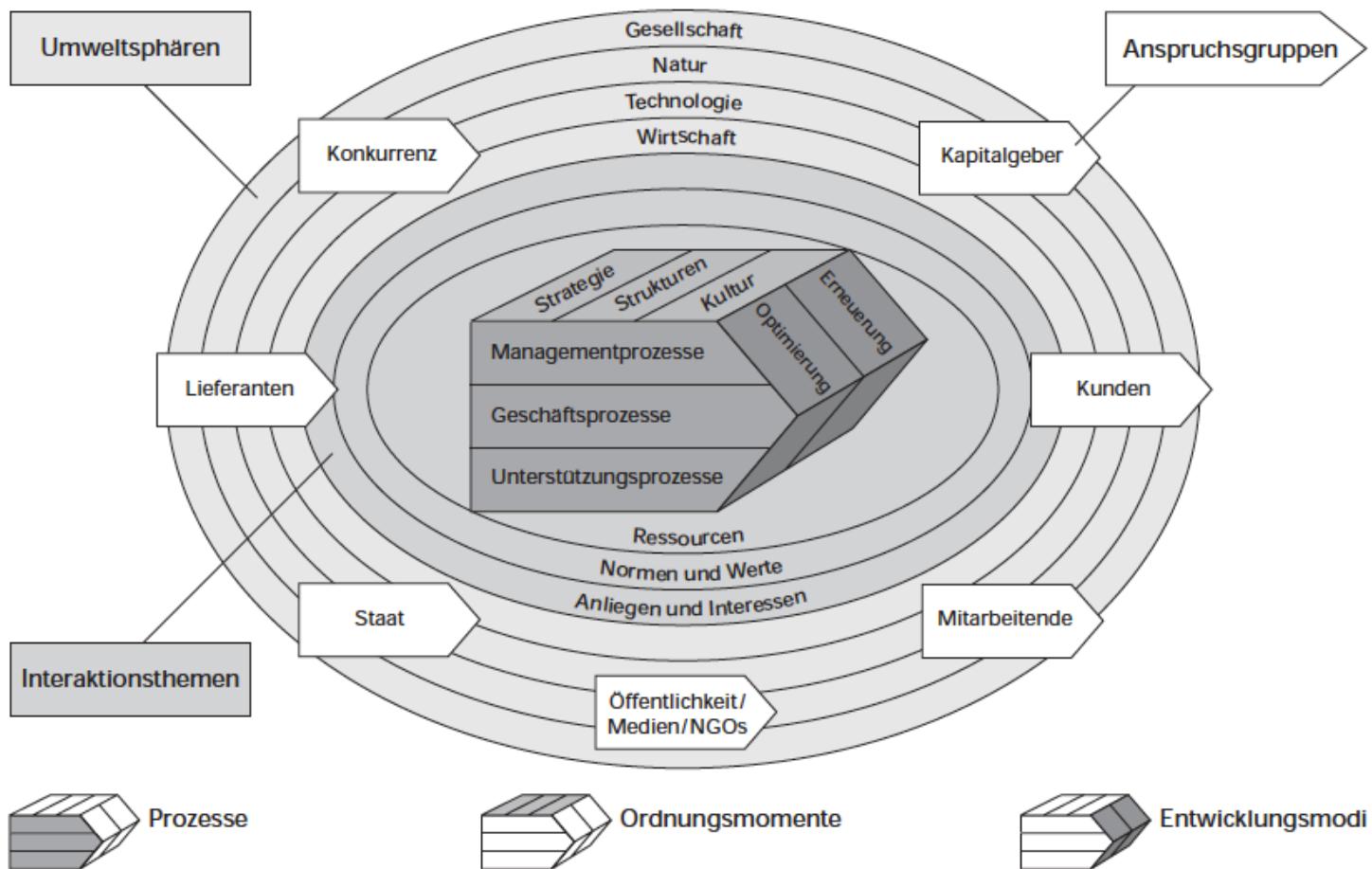
## Das Unternehmen als System



Quelle: Systemanalyse; [www.daswirtschaftslexikon.com](http://www.daswirtschaftslexikon.com)

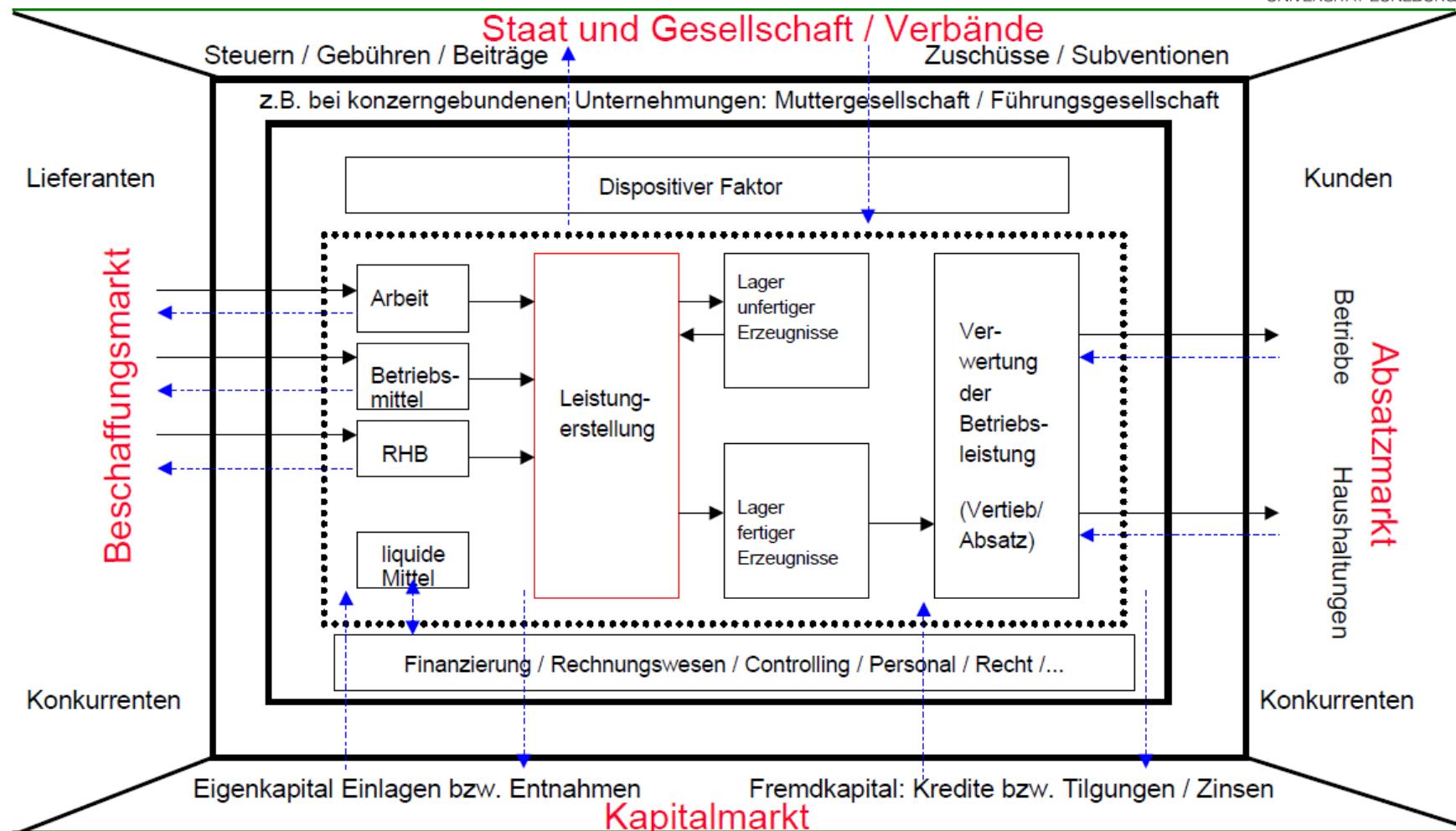
<http://www.daswirtschaftslexikon.com/d/systemanalyse/systemanalyse.htm>; Zugriff: 04.06.2019

# Warum? - BWL und System (3)



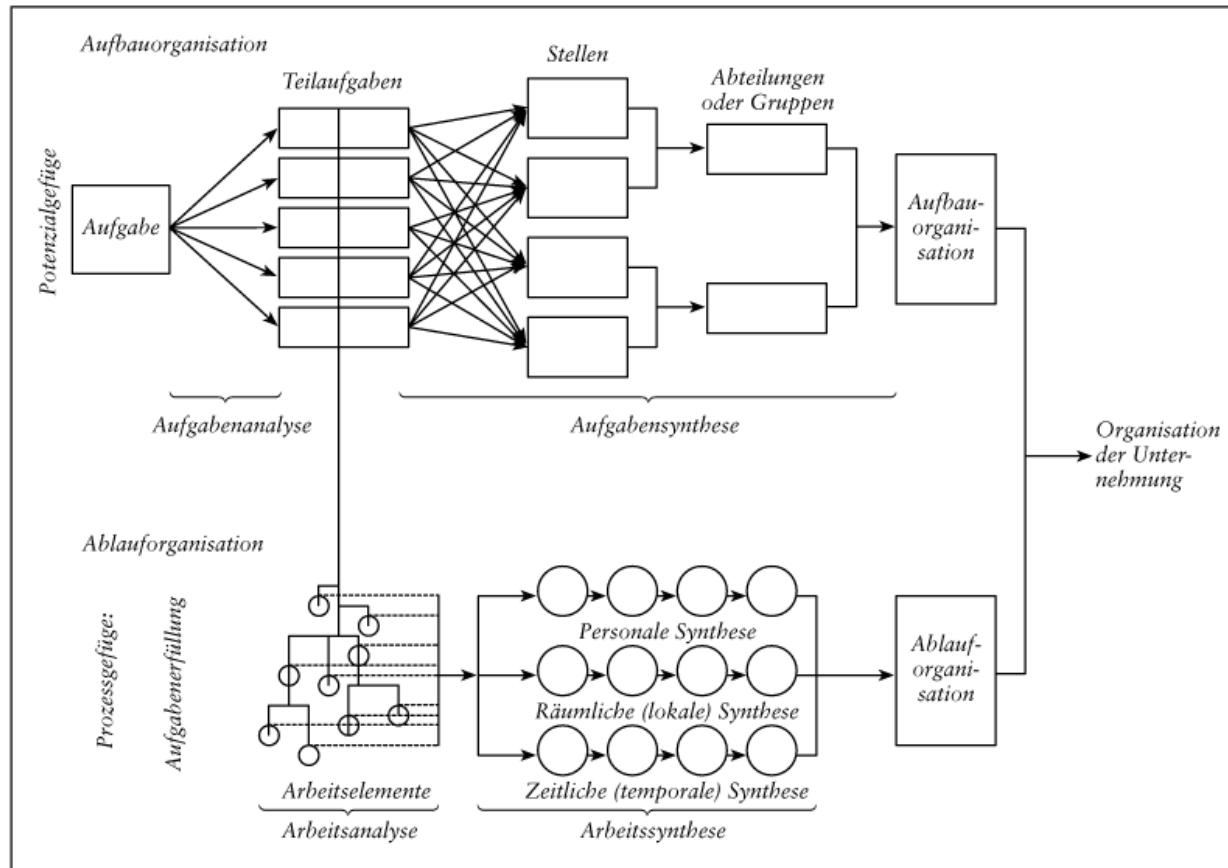
Quelle: <http://www.hagen-management.at/wp-content/uploads/2010/11/sgmm.png>; Zugriff am 14.06.2013

# Warum? - BWL und System (4)



Quelle: <http://www.klauk.de/pdf/bwl0505.pdf>; Zugriff am 12.11.2012

## Beispiel Organisation



Aufbauorganisation

=

Struktur

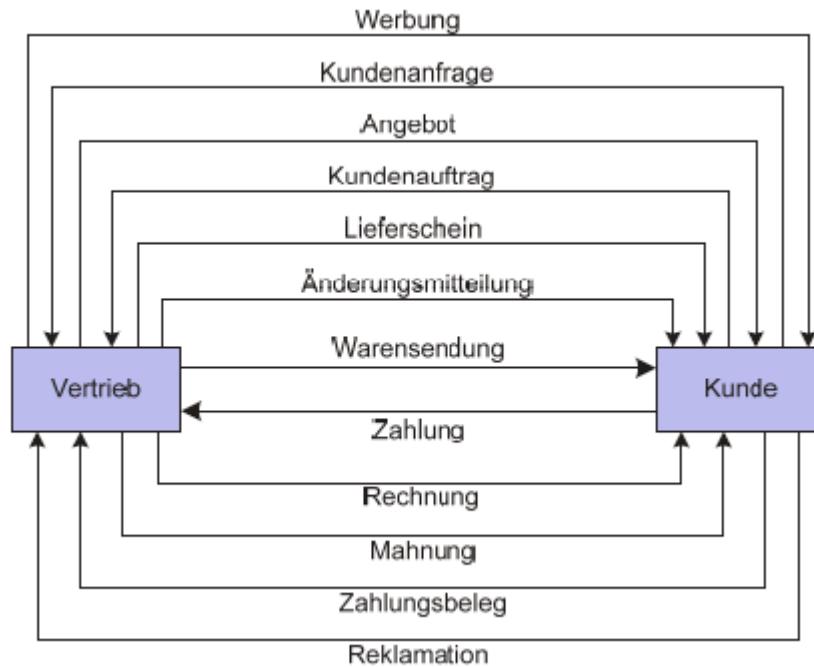
Ablauforganisation

=

Verhalten (Prozess)

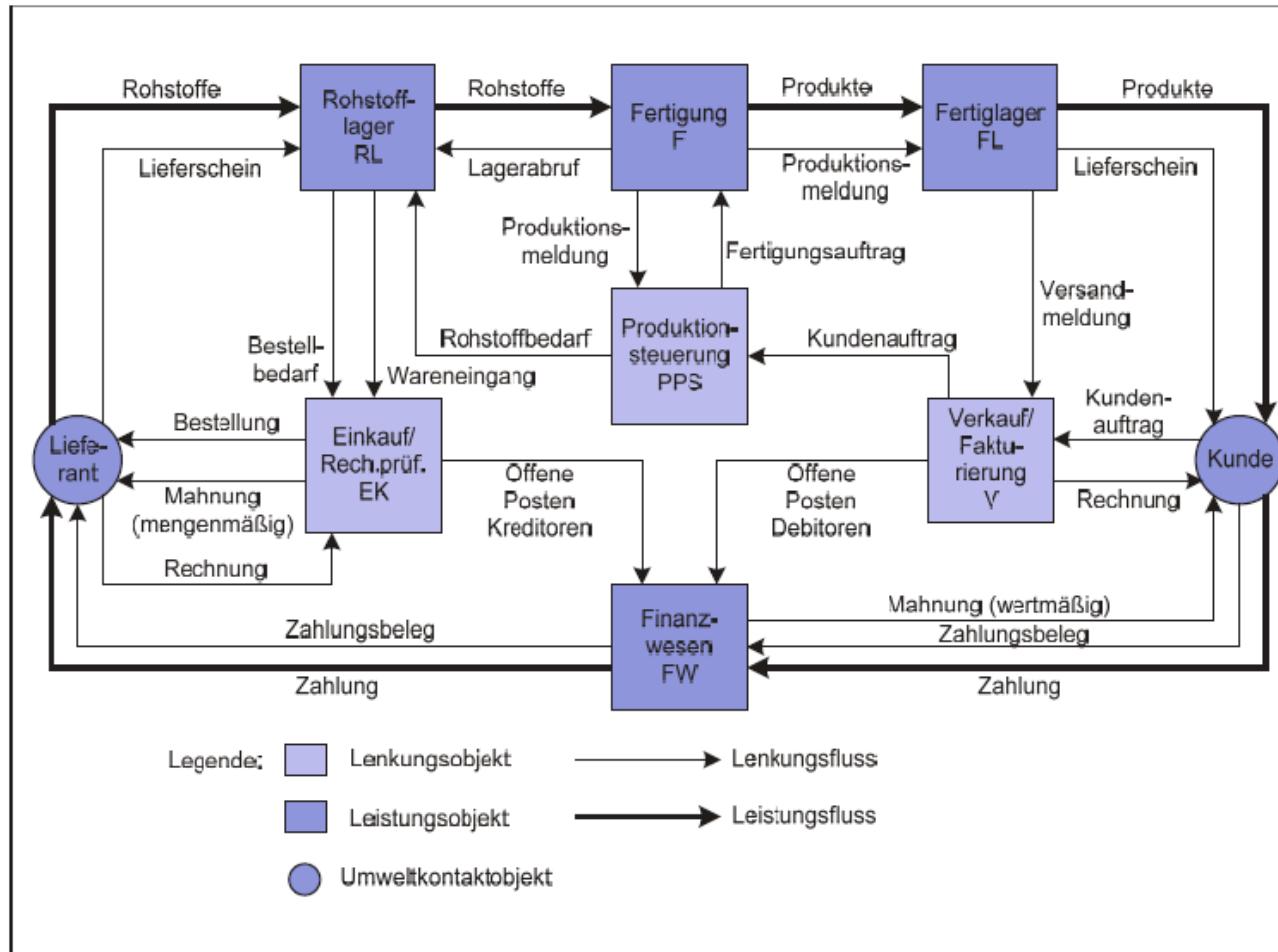
Quelle: <http://www.daswirtschaftslexikon.com/abbildungen/96-aufbau- und ablauforganisation.gif>; Zugriff: 23.11.2012

## Austausch zwischen Systemen



Quelle: Ferstl, O., Sinz, E., Einführung in die Wirtschaftsinformatik, 7.Aufl., Oldenbourg, 2015

## Austausch zwischen Systemen



Quelle: Ferstl, O., Sinz, E., Einführung in die Wirtschaftsinformatik, 7.Aufl., Oldenbourg, 2015

„A complex system is made up of a large variety of components or elements that possess specialized functions.

These elements are organized in internal hierarchical levels (in the human body, for example, cells, organs, and systems of organs).

The different levels and individual elements are linked by a great variety of bonds. Consequently there is a high concentration of interconnections.

The interactions between the elements of a complex system are of a particular type; they are nonlinear interactions.“

*de Rosny, N., The Macroscope, 1975*

„complexity characterizes something with many evolving parts that interact with each other in various ways, displaying nonlinear patterns in the aggregate, which, often, is not additive. This something usually is called a ‘complex system’.” Put differently, complex systems satisfy the following characteristics:

- Cardinality: many parts (from particles to agents) constitute the system
- Diversity: parts are different from one another
- Dimensionality: parts differ from one another in multiple ways, i.e. along several dimensions
- Connectivity: parts act, interact, and adapt through networks
- Nonlinearity: the relationship between variables is often nonlinear and the aggregate of the parts does not equal the sum of their characteristics or actions.“

<https://eageneva.org/blog/2018/10/18/an-introduction-to-complexity-science-for-social-sciences>; Zugriff: 05.05.2023

# Warum? - System und Komplexität (2)

„a complex system includes multiple interacting components forming a whole irreducible to its parts.“

*Sabzian, H. et al., Theories and Practice of Agent based Modeling: Some practical Implications for Economic Planners, 2019*

„A system is complex, when it can be represented efficiently by different models at different scales.“

Fromm, J., The emergence of complexity, 2004

“a complex system is complex on many scales”

“a complex system is formed out of a hierarchy of interdependent subsystems”

Fromm, J., The emergence of complexity, 2004

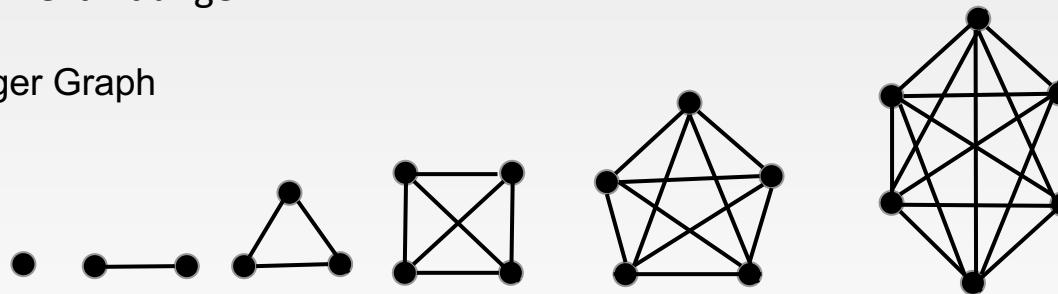
„complexity is a measure related to the degree to which a system has both a multiplicity of components (subsystems) at any given level of organization (below) and the number of such levels forming a hierarchy.“

Mobus, G., Systems Science: Theory, Analysis, Modeling, and Design, 2021

## Struktur

- Anzahl der Elemente
- Anzahl der Verbindungen

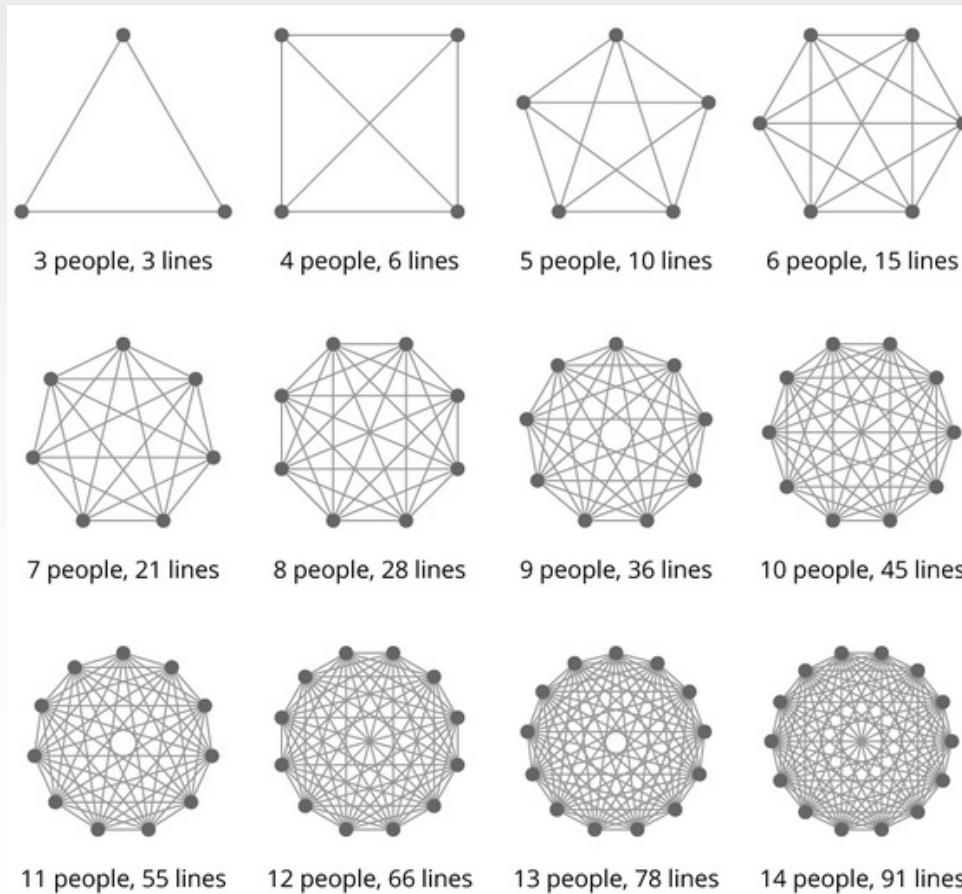
Vollständiger Graph



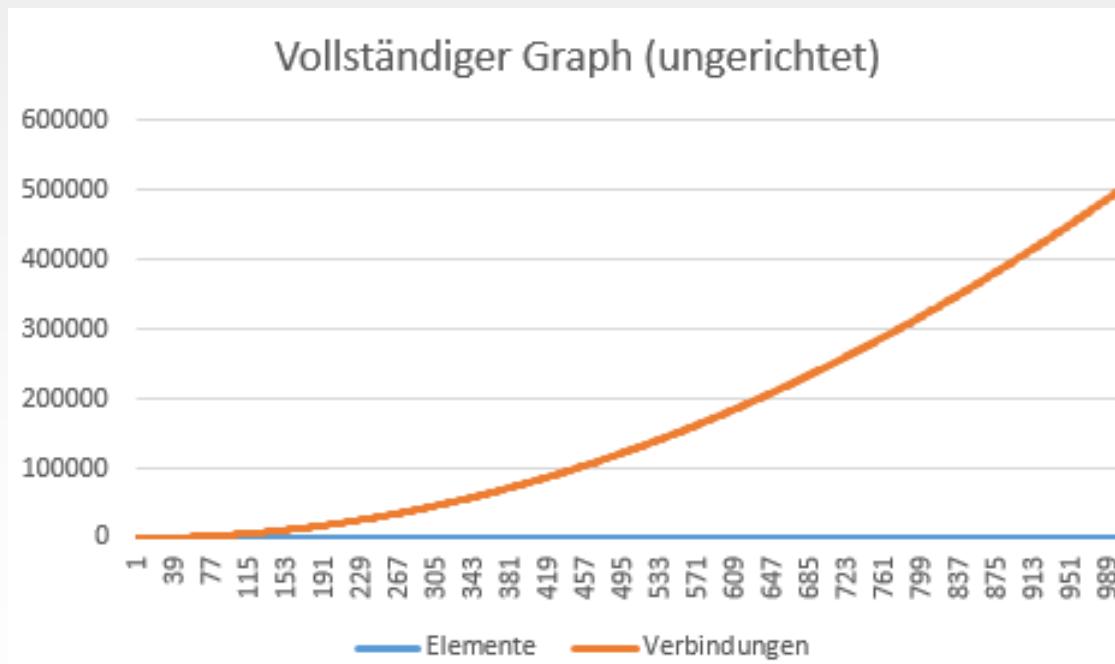
Elemente	1	2	3	4	5	6
Verbindungen	0	1	3	6	10	15

$$\text{Dreieckszahl } \triangle_n = \frac{n * (n-1)}{2}$$

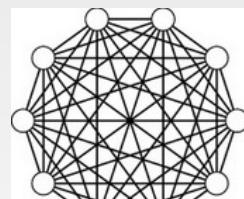
# Warum? - System und Komplexität (4)



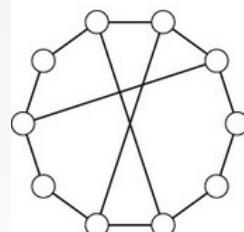
Quelle: Developing Leaders: What To Do When Your Team Grows Too Big  
<https://getlighthouse.com/blog/developing-leaders-team-grows-big/>; Zugriff: 03.07.2018



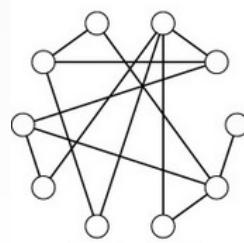
Architektur von  
Informations-  
systemen



„zu regide“  
(starr)



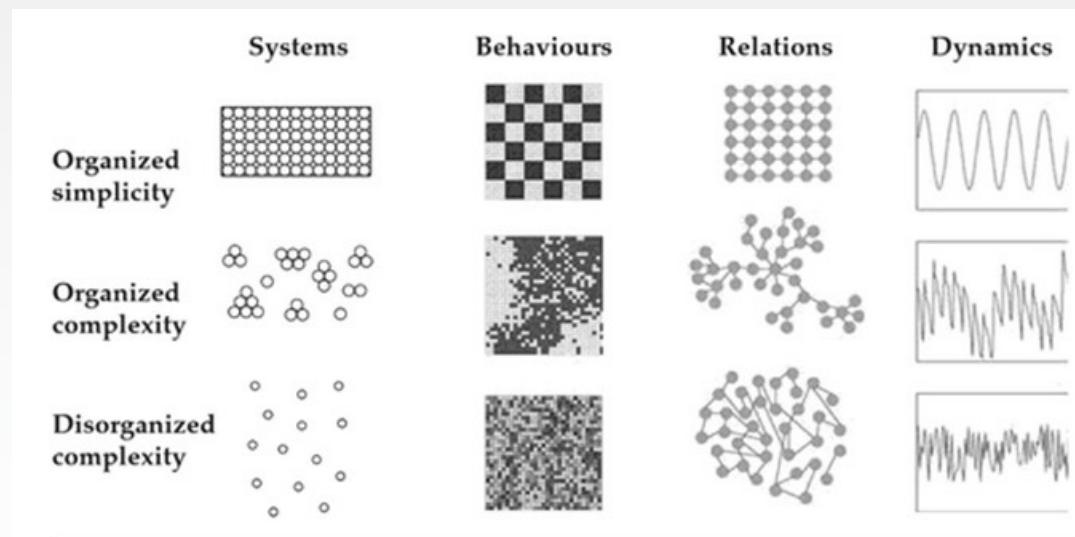
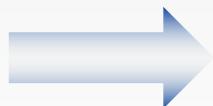
adaptiv  
(komplex)



„zu willkürlich“  
(chaotisch)

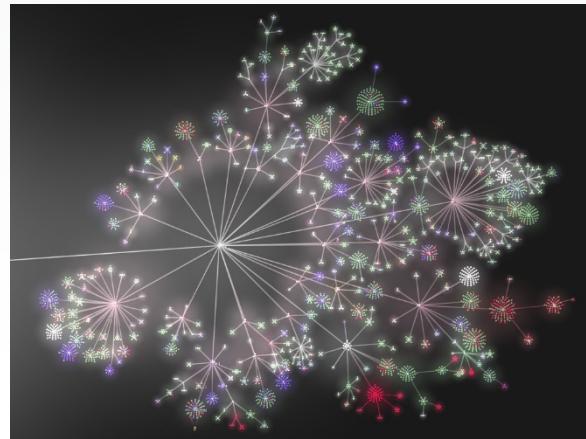
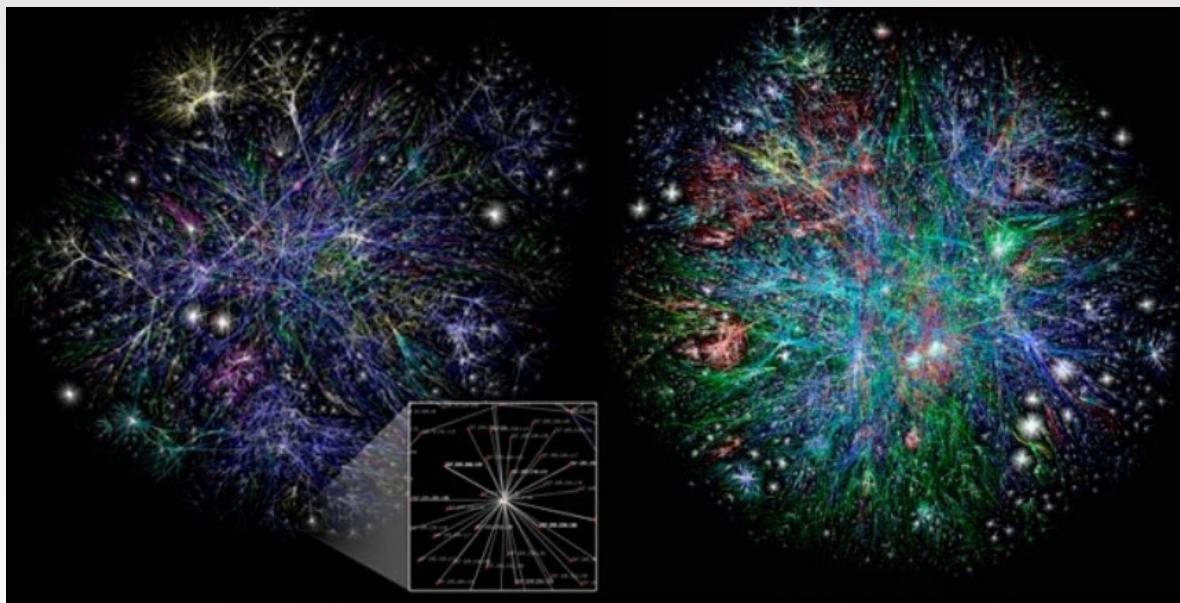
# Warum? - System und Komplexität (7)

Architektur von  
Informations-  
systemen



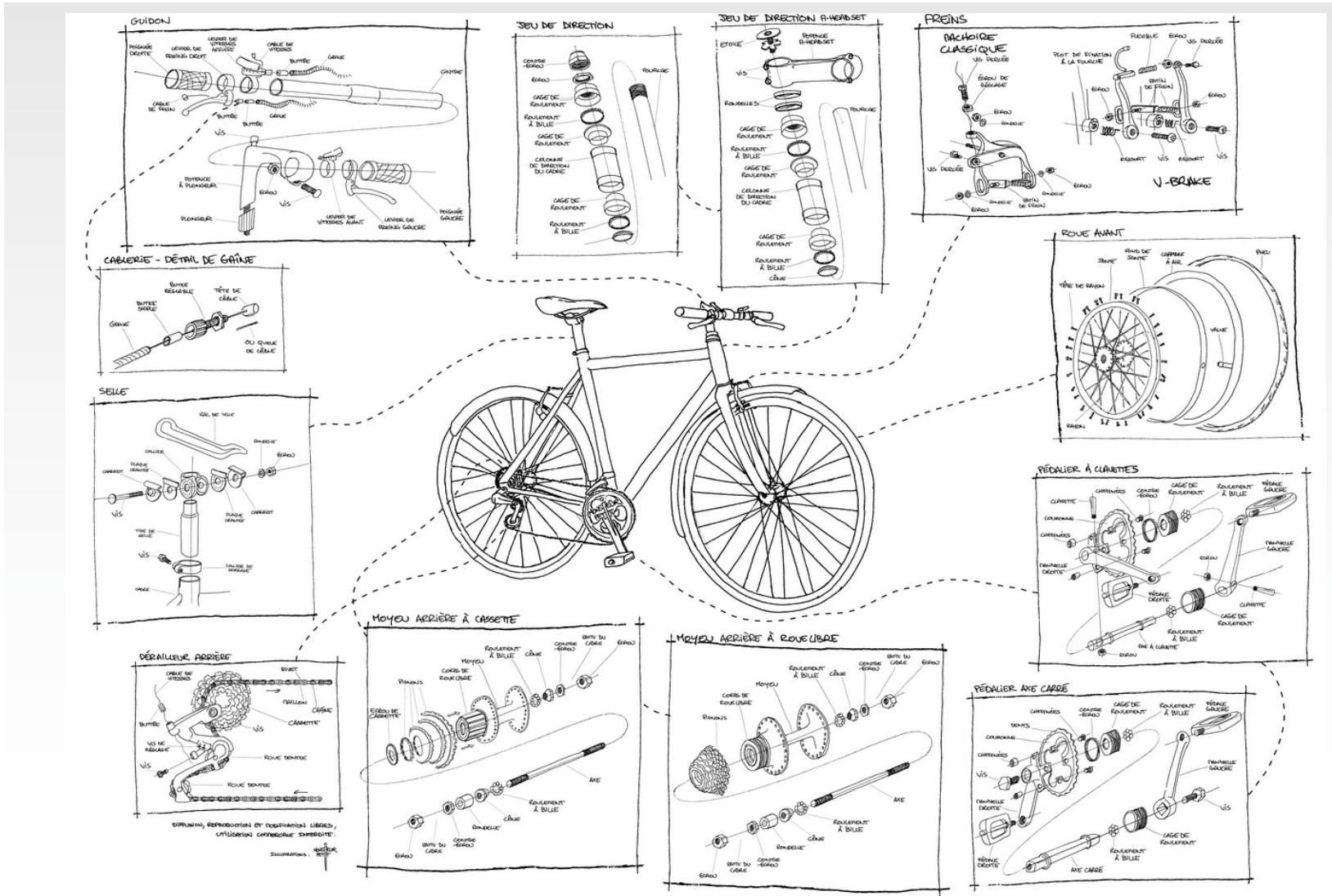
Ramalingam, B., Aid on the Edge of Chaos: Rethinking International Cooperation in a Complex World, 2014

# Warum? - System und Komplexität (8)



Quelle: <https://www.pinterest.de/pin/270919733804505261/>;  
<http://source.googlecode.com/svn/trunk/gource-linux.jpg>; Zugriff: 24.11.2017  
© Thomas Slotos

# Warum? - System und Komplexität (9)



Quelle: [http://wiklou.org/images/c/c1/Dessin\\_technique\\_isometrique\\_-\\_Vue\\_eclatée\\_vélo\\_-\\_sous-titrage\\_français\\_HD.jpg](http://wiklou.org/images/c/c1/Dessin_technique_isometrique_-_Vue_eclatée_vélo_-_sous-titrage_français_HD.jpg); Zugriff: 14.10.2015

## Warum gibt es organisierte Komplexität ?

- Systeme, die miteinander in Beziehung treten, konkurrieren häufig miteinander. Ändert sich ein System muss sich das andere System anpassen um funktionsfähig zu bleiben. Aus Sicht eines Systems führen neue Anforderungen aus der Systemumgebung dazu, dass die Funktionsfähigkeit des Systems nur durch geänderte strukturell/verhaltensmäßige Anpassung erfolgen kann. Diese Anpassung geschieht häufig in Form von zusätzlichen Strukturen/Prozessen innerhalb des Systems. Dies ist ein fortwährender und evolutionären Vorgang, so dass immer komplexere Systeme entstehen.

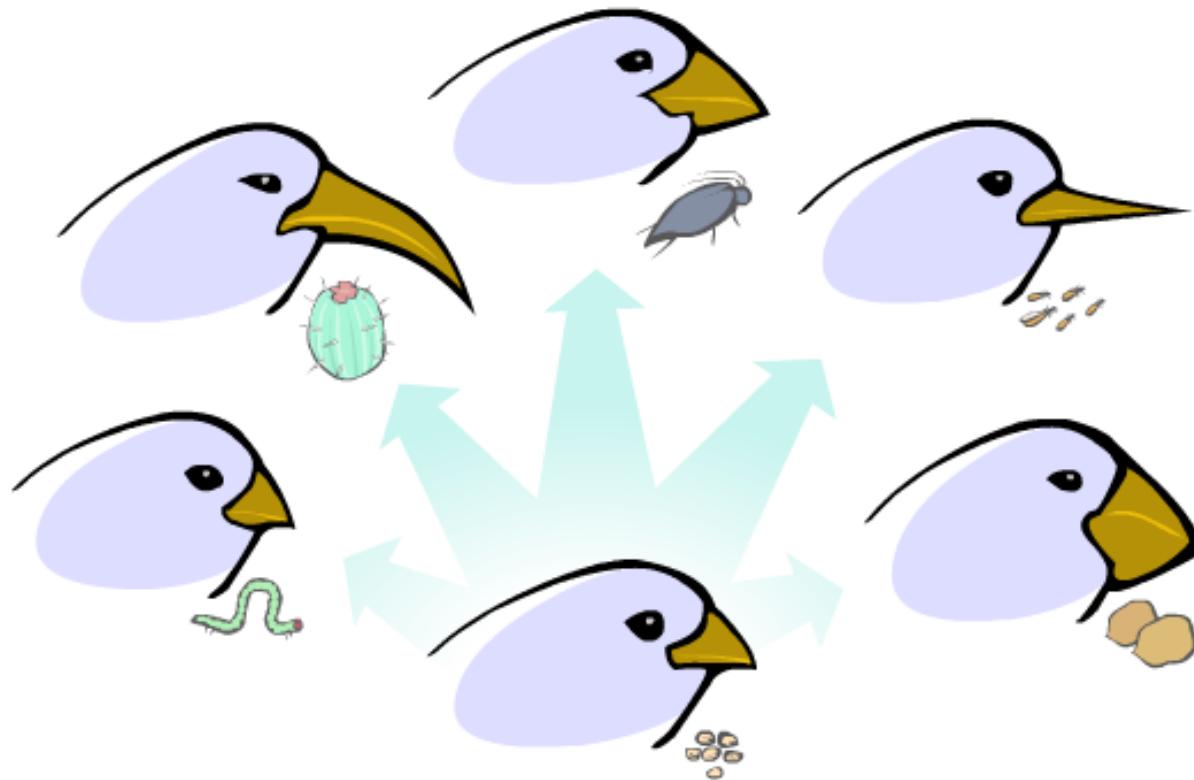
## Wie entstehen organisiert komplexe Systeme ?

- Organisiert komplexe Systeme entstehen durch Spezialisierung und Differenzierung von Funktionen des Systems. Funktionen werden durch das Zusammenwirken der im System vorhandenen Strukturen und den darauf arbeitenden Prozessen bereitgestellt. Die Gesamtfunktionalität eines Systems ergibt sich aus den Teifunktionalitäten des Systems und deren Zusammenwirken. Die Gesamtfunktionalität entsteht Bottom-up in dem einfachere Funktionseinheiten zu komplexeren zusammengefügt werden.

## Wie können organisiert komplexe Systeme untersucht werden?

- Die Verwendung von generellen Prinzipien, Methoden und Vorgehensweise bietet die Möglichkeit organisiert komplexe System zu untersuchen.

# Warum? - System und Adaptivität (1)



Quelle: <http://wisebrain.info/wp-content/uploads/2013/05/adaptive.png>; Zugriff: 09.11.2016

„An adaptive system is a set of interacting or interdependent entities, real or abstract, forming an integrated whole that together are able to respond to environmental changes“

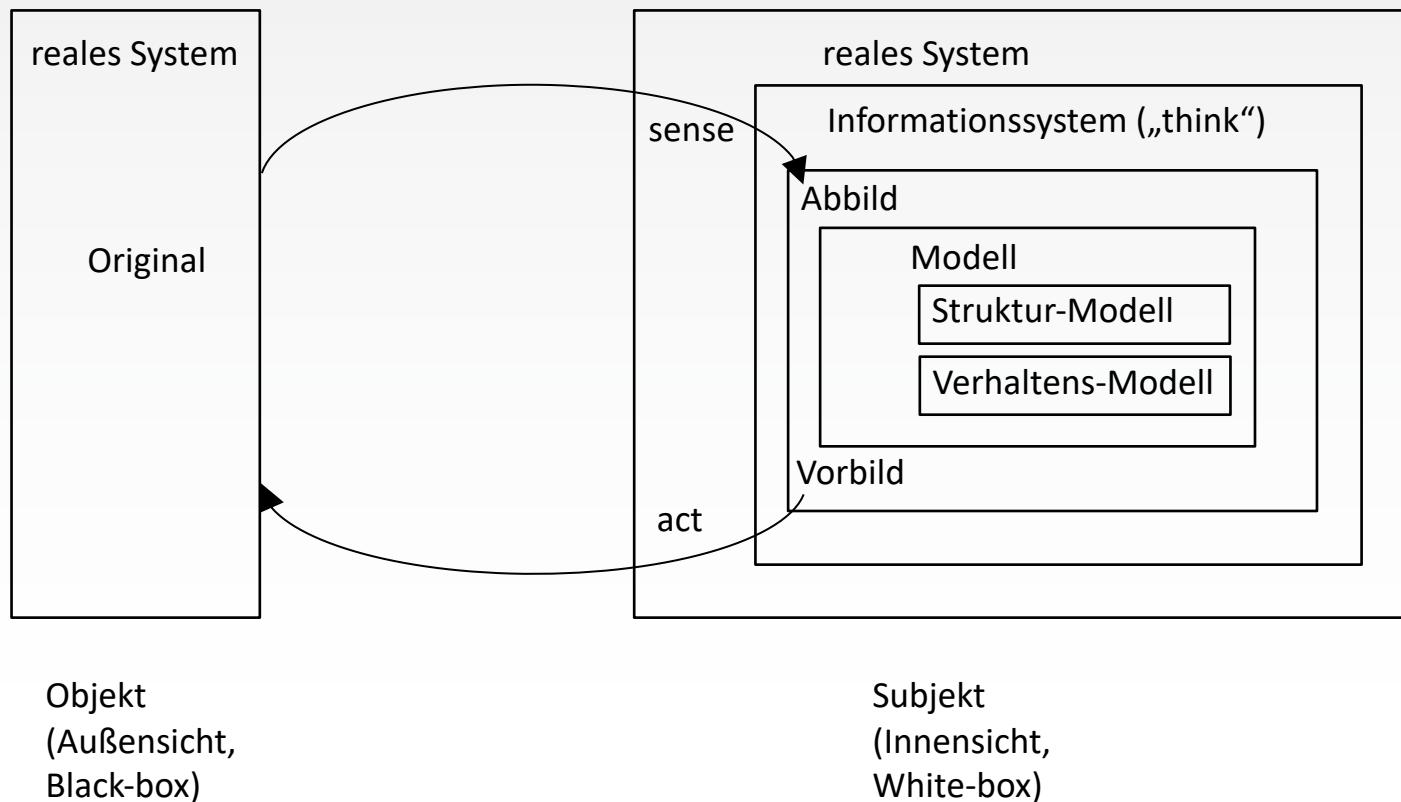
[https://en.wikipedia.org/wiki/Adaptive\\_system](https://en.wikipedia.org/wiki/Adaptive_system); Zugriff: 05.05.2023

“An adaptive system ... is a system that changes its behavior in response to its environment. The adaptive change that occurs is often relevant to achieving a goal or objective.“

<https://necsi.edu/adaptive>; Zugriff: 05.05.2023

„All adaptive behavior rests on the circular processing of information between the organism and its environment.“

Quelle: Fuster, J.; The Prefrontal Cortex – An Update: Time Is of The Essence; 2001  
[https://www.cell.com/neuron/pdf/S0896-6273\(01\)00285-9.pdf](https://www.cell.com/neuron/pdf/S0896-6273(01)00285-9.pdf); Zugriff: 01.08.2019



Um in einer Umwelt zu bestehen/überleben, muss sich ein System den Anforderungen aus der Umgebung anpassen. Die Anpassung wird Adaptivität genannt.

Generell lässt sich zwischen struktureller und verhaltensmäßiger Adaptivität unterscheiden.

- Verhaltensmäßige Adaptivität

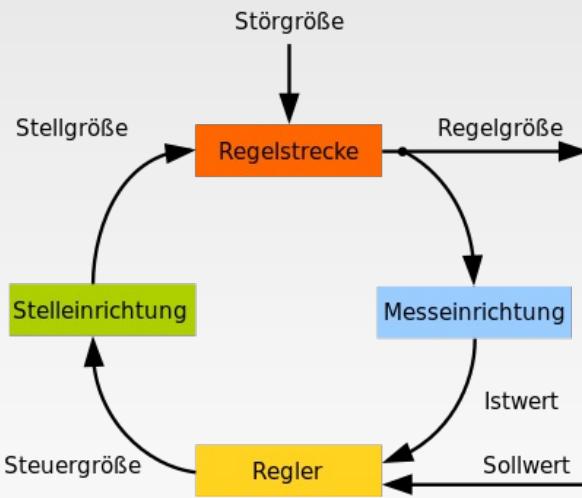
Wird bei Messung (Ist-Werte) von systemrelevanten Faktoren eine Abweichung des Soll-Wertes festgestellt, muss das System bestehende Prozesse durch veränderte Aktivitäten anpassen (z.B. Beschleunigung oder Parallelisierung der Prozesse).

- Strukturelle Adaptivität

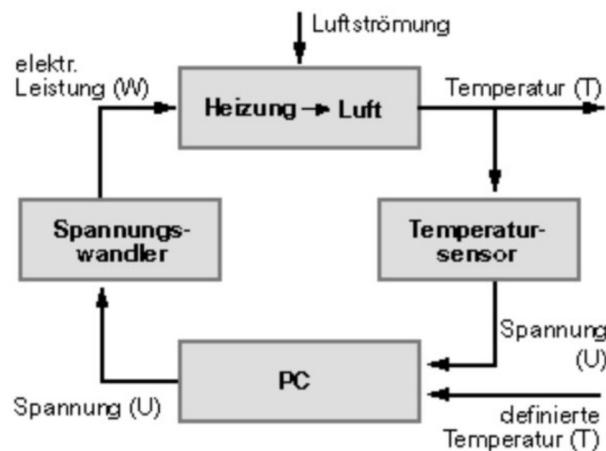
Sollte die Anpassung der Prozesse auf der bestehenden Struktur nicht ausreichen, dann kommt es zu einer Reorganisation der internen Struktur des Systems bei der nicht mehr benötigte Elemente/Beziehungen wegfallen und wichtige neue Elemente/Beziehungen hinzugefügt werden.

In der Softwaretechnik findet die Anpassung durch die SoftwareentwicklerInnen während der „Wartung“ (Maintainance) des Softwaresystems statt.

## Verhaltensmäßige Adaptivität

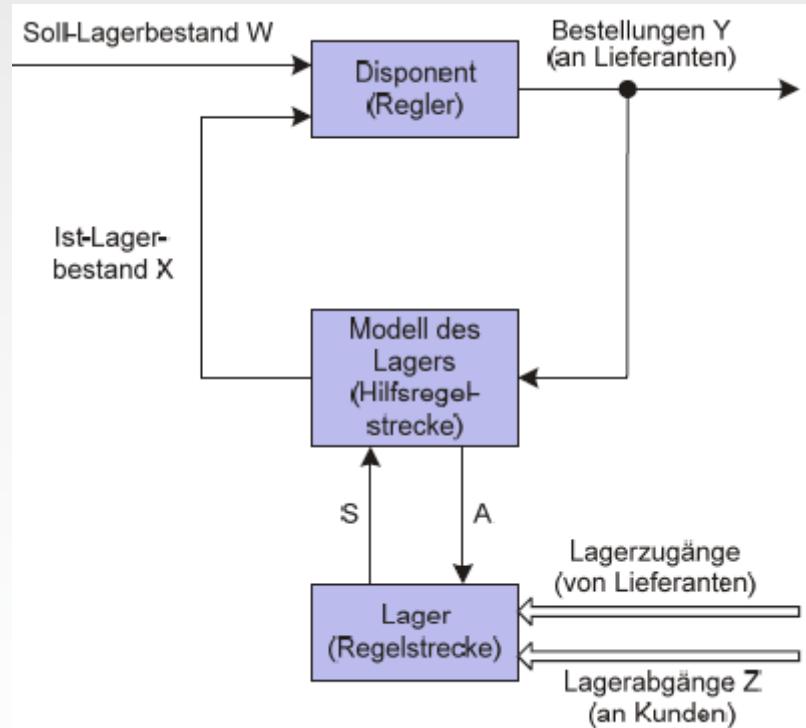


<https://de.wikipedia.org/wiki/Temperaturregler>; Zugriff: 17.10.2018



Hofstetter, J. et.al.; Blockdiagramme für die Software-Systemmodellierung - Ein praxisorientierter Leitfaden, 2020

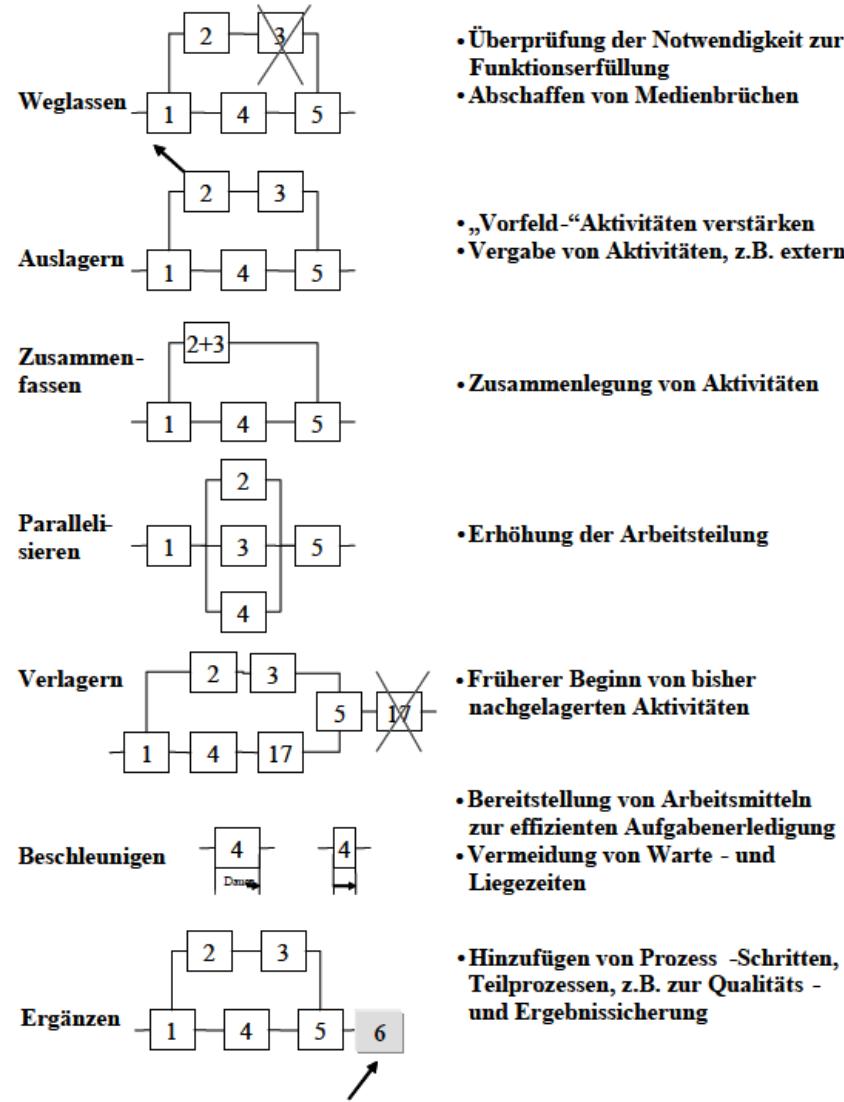
## Verhaltensmäßige Adaptivität



Quelle: Ferstl, O., Sinz, E.: Grundlagen der Wirtschaftsinformatik, 7.Aufl., Oldenbourg, 2015

## Verhaltensmäßige Adaptivität

Geschäftsprozessoptimierung  
- durch Prozessrestrukturierung



- Überprüfung der Notwendigkeit zur Funktionserfüllung
- Abschaffen von Medienbrüchen

- „Vorfeld-“Aktivitäten verstärken
- Vergabe von Aktivitäten, z.B. extern

- Zusammenlegung von Aktivitäten

- Erhöhung der Arbeitsteilung

- Früherer Beginn von bisher nachgelagerten Aktivitäten

- Bereitstellung von Arbeitsmitteln zur effizienten Aufgabenerledigung
- Vermeidung von Warte- und Liegezeiten

- Hinzufügen von Prozess-Schritten, Teilprozessen, z.B. zur Qualitäts- und Ergebnissicherung

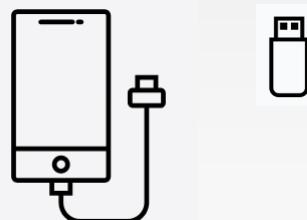
## Strukturelle Adaptivität

Muster (Pattern)



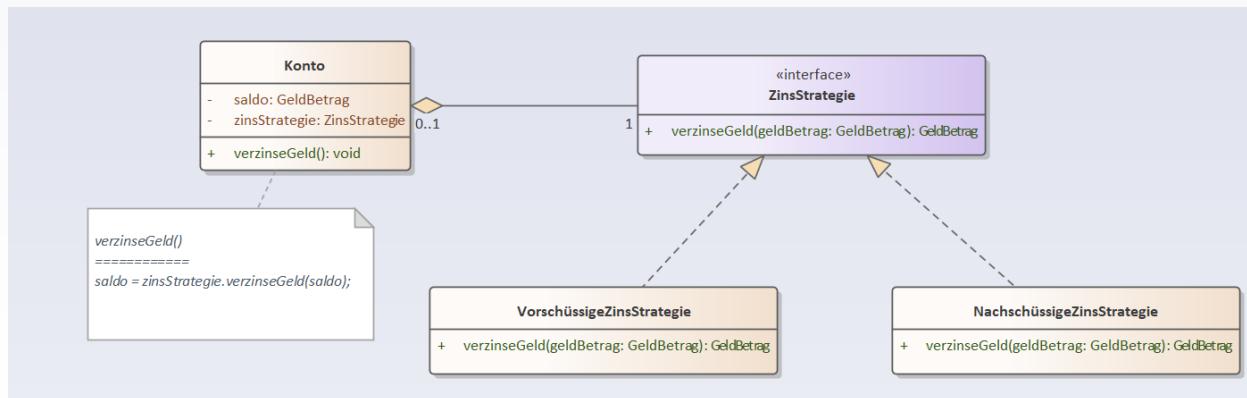
Common  
Was? / Fest

Hardware (USB-Interface)



Variable  
Wie? / Flexible

Software (Strategy pattern)



Common  
Was? / Fest

Variable  
Wie? / Flexible

[https://de.freepik.com/freie-ikonen/usb-stick-beschrieben\\_742590.htm](https://de.freepik.com/freie-ikonen/usb-stick-beschrieben_742590.htm);  
<https://thenounproject.com/icon/cell-phone-with-usb-cable-2025259/>  
<https://de.depositphotos.com/57924313/stock-illustration-laptop-icon.html>

Zugriff: 06.05.2022

## Warum gibt es Adaptivität?

- Systeme, die miteinander in Beziehung treten, konkurrieren häufig miteinander. Ändert sich ein System muss sich das andere System anpassen um funktionsfähig zu bleiben. Aus Sicht eines Systems führen neue Anforderungen aus der Systemumgebung dazu, dass die Funktionsfähigkeit des Systems nur durch geänderte strukturell/verhaltensmäßige Anpassung erfolgen kann. Diese Anpassung geschieht häufig in Form von zusätzlichen Strukturen/Prozessen innerhalb des Systems. Dies ist ein fortwährender und evolutionären Vorgang, so dass innere Strukturen/Prozesse sich fortlaufend ändern.

## Wie entstehen adaptive Systeme ?

- Adaptive Systeme entstehen durch Aufbau von internen Strukturen/Prozessen die dazu dienen die Funktion (Aufgabe) des Systems zu erfüllen. Um zu überleben muss daher das System kontinuierlich die Umwelt wahrnehmen und bei geänderten Umweltfaktoren (Anforderungen) interne Strukturen/Prozesse anpassen. Im Extremfall kann sich auch die Funktion (Aufgabe) stark ändern.

## Wie können adaptive Systeme untersucht werden?

- Die Verwendung von Prinzipien der Regelungstechnik erlauben die veränderten Zustände eines Systems zu erfassen und zu ändern.

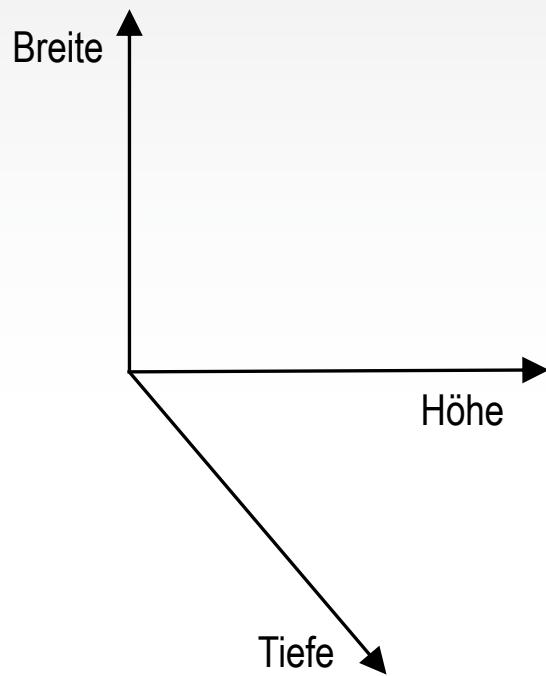
# Wo?

Der physische Raum kann in die Dimensionen Höhe, Breite und Tiefe unterteilt werden. Alle 3 Dimensionen sind disjunkt bzw. orthogonal von/zu einander, d.h. keine Dimension beeinflusst die andere obwohl ein Zusammenhang zwischen den Dimensionen besteht.

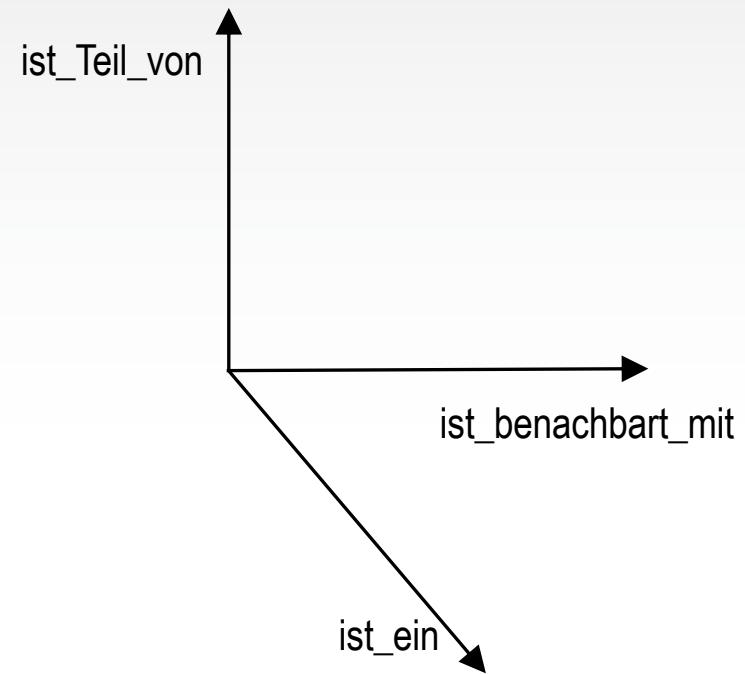
Um die Anwendungsdomäne zu analysieren und zu dokumentieren für die ein Informationssystem entwickelt wird, können auch 3 mentale, logische und orthogonale Dimensionen unterschieden werden. Diese werden nach den statischen Beziehungsarten zwischen den Elementen benannt. Eine Dimension ist die „ist\_Teil\_von“-Beziehung, die angibt, dass eine Einheit Teil eines größeren Einheit ist (z.B. sind Vorlesungsteilnehmer und Vorlesungsinhalt Teil der Vorlesung). Eine andere Dimension ist die „ist\_ein“-Beziehung, die aufzeigt, dass eine Einheit Eigenschaften aufweist, die auch in verallgemeinerten Einheiten vorkommen (z.B. sind Student und Dozent Vorlesungsteilnehmer). Eine weitere Dimension ist die „ist\_benachbart\_mit“-Beziehung, die angibt, dass 2 Einheiten auf gleicher Ebene verbunden sind. ( z.B. sind Student und Vorlesungsinhalt dahingehend miteinander verbunden, dass der Student den Vorlesungsinhalt lernt).

Bei der „ist\_Teil\_von“-Beziehung und der „ist\_ein“-Beziehung sind die Einheiten auf unterschiedlichen Ebenen vorhanden. Beide Beziehungen behandelt Über- bzw. Unterordnung von Einheiten. Die „ist\_benachbart\_mit“-Beziehung behandelt Einheiten gleicher Ebene und zeigt damit die Gleichordnung der Einheiten an.

## Räume – physischer und mentaler/konzeptueller/logischer Raum



**Physischer Raum**

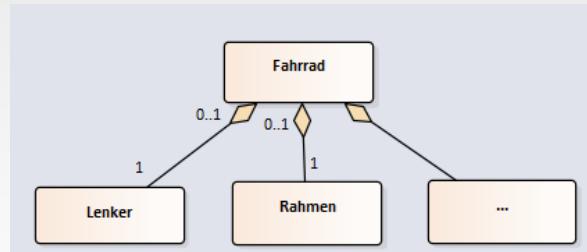


**Mentaler/  
Konzeptueller/  
Logischer Raum**

# Wo? - Dimensionen im Raum (3)

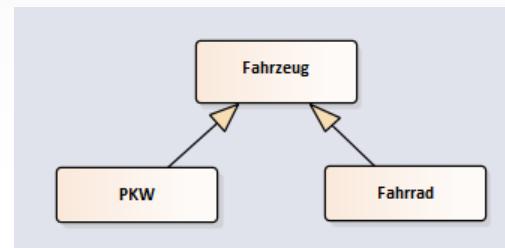
## Dimensionen im logischen Raum (in UML)

Dimension „ist\_Teil\_von“  
(partonomic relation)  
- Aggregation



- Seitenansicht
- Über-/Unterordnung
- Hierarchy (Partonomy)
- ↑ logische AND-Verknüpfung
- Viele Objekte

Dimension „ist\_ein(e\_Art\_von)“  
(taxonomic relation)  
- Generalisierung



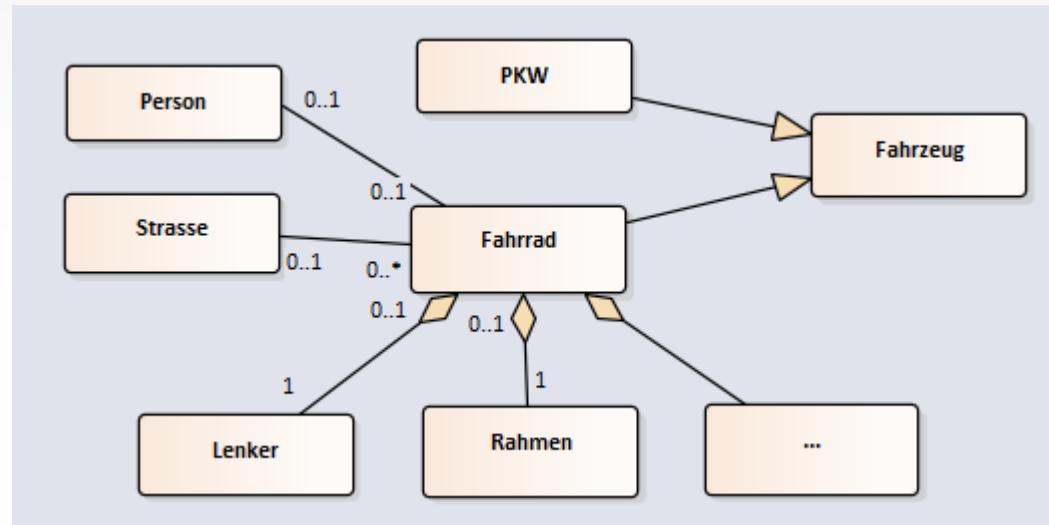
- Seitenansicht
- Über-/Unterordnung
- Hierarchy (Taxonomy)
- ↓ logische XOR-Verknüpfung
- Ein Objekt

Dimension „ist\_benachbart\_mit“  
(thematic relation)  
- Assoziation



- Draufsicht
- Gleichordnung
- Hierarchy (Schema)
- ↔ logische AND-Verknüpfung
- Viele Objekte

## Integration der 3-Dimensionen des logischen Raumes in einem UML-Diagramm (2-dimensional)



Um ein komplexes System darzustellen, können 2 unterschiedliche Arten von Abstraktionshierarchien benutzt werden. Eine Abstraktionshierarchie ist eine baumartige Struktur bei der die Blätter die speziellen und die Wurzel die allgemeinen Eigenschaften eines Systems darstellen.

Als Abstraktionsmechanismus kann entweder die „**Aggregation**“ („**ist\_Teil\_von-Beziehung**“) oder die „**Generalisierung/Spezialisierung**“ („**ist\_ein(e\_Art\_von-Beziehung**“) verwendet werden. Wird die „Aggregation“ bzw. „partonomic relation“ verwendet, dann entsteht eine Struktur, die sich „**Partonomie**“ nennt; wird die „Generalisierung/Spezialisierung“ bzw. „taxonomic relation“ verwendet, dann entsteht eine Struktur, die sich „**Taxonomie**“ nennt. Die „Partonomie“ stellt eine logische **AND-Beziehung** und die „Taxonomie“ eine logische **XOR-Beziehung** zwischen den Einheiten der Hierarchie her.

Beiden Hierarchiearten ist gemeinsam, dass das Durchlaufen eines Pfades von einem Blatt zur Wurzel „**Abstraktion**“ und, dass das Durchlaufen eines Pfades von der Wurzel zu einem Blatt „**Konkretisierung**“ genannt wird.

„When classes are systematically related to each other by means of subsumption, we arrive at a taxonomy; and when a whole of a certain kind is systematically and exhaustively partitioned, we arrive at a partonomy.“

Quelle: Johansson, I., Lynoe, N.; Taxonomy, Partonomy, and Ontology; in: Medicine & Philosophy, 2008

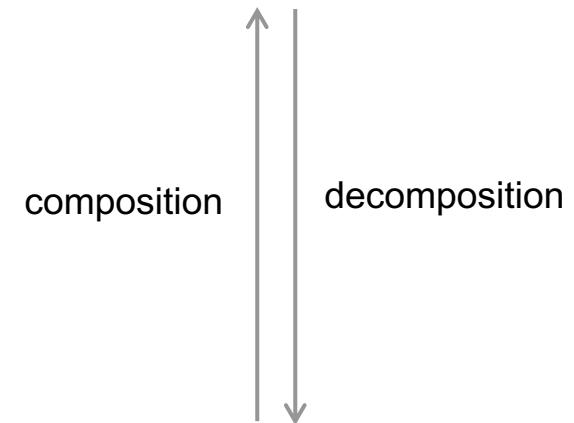
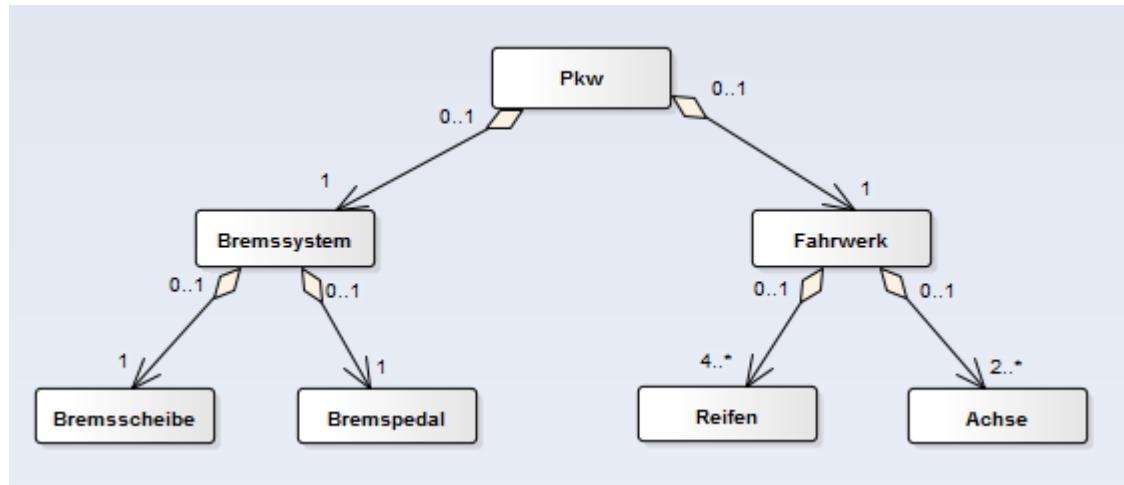
„... two general forms of organization of knowledge, taxonomic – that is, subdivision into kinds, and partonomic – that is, subdivision into parts.

Both forms of organization ... produce hierarchies based on transitive, asymmetric relations.“

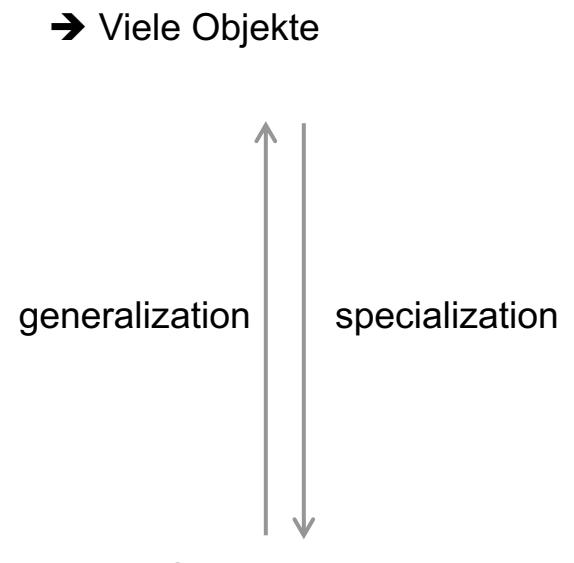
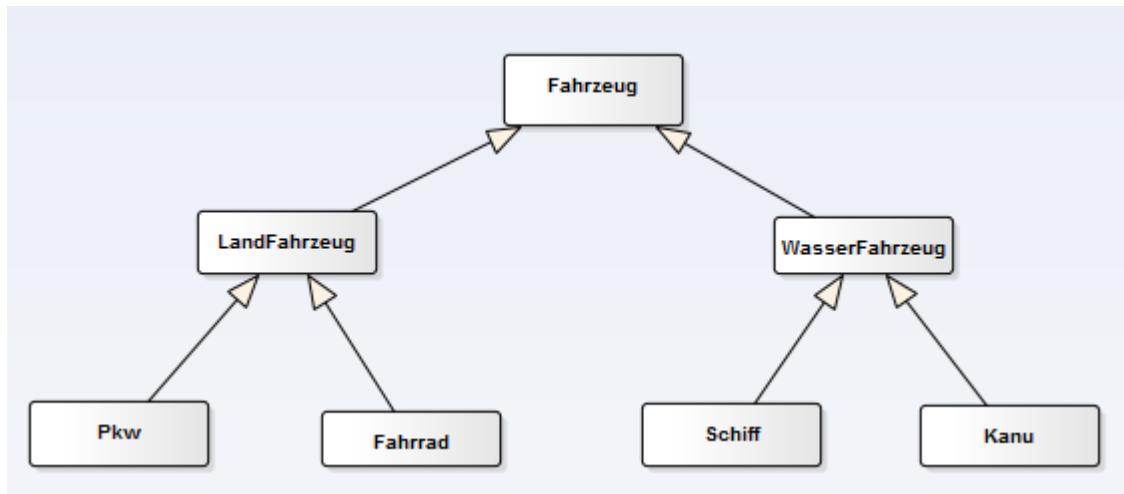
Quelle: Tversky, B.; Where partonomies and taxonomies meet, 1990

# Wo? - Hierarchie und Abstraktion (3)

Partonomie (is-part-of hierarchy – Und-Beziehung)

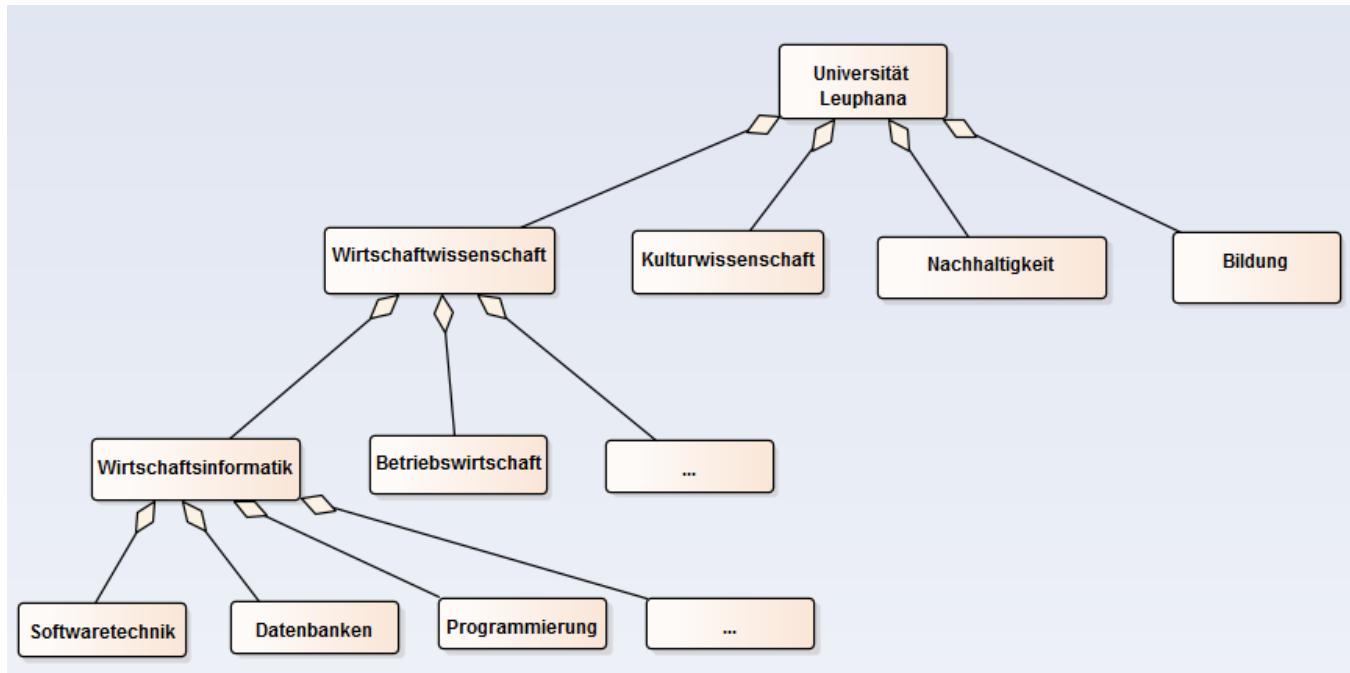


Taxonomie (is-a hierarchy – exklusive Oder-Beziehung)



# Wo? - Hierarchie und Abstraktion (4)

Partonomie (is-part-of hierarchy) – Virtuelles System und Ebenen



Organisationsebene

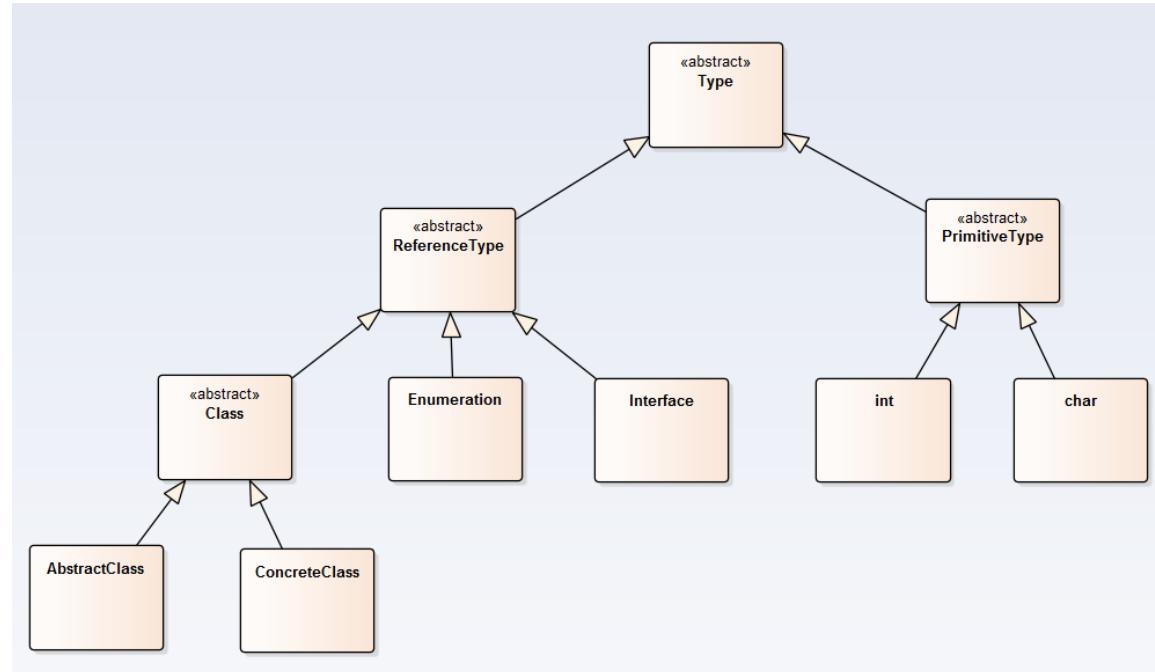
Fakultätsebene

Studiengangsebene

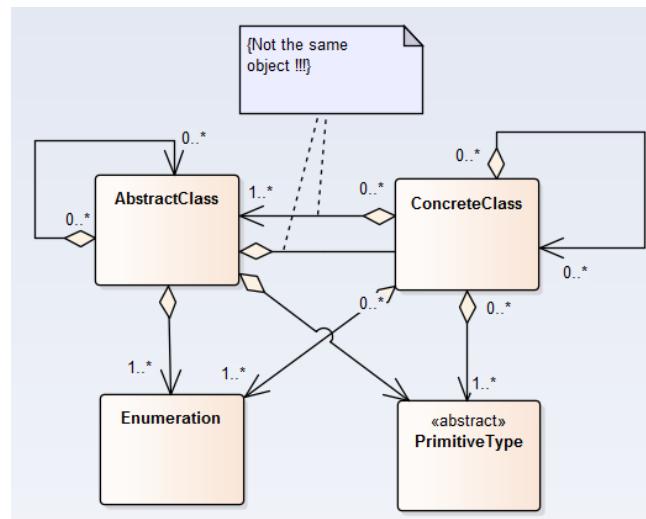
Vorlesungsebene

# Wo? - Hierarchie und Abstraktion (5) – Typen in Java

Taxonomie (is-a hierarchy)

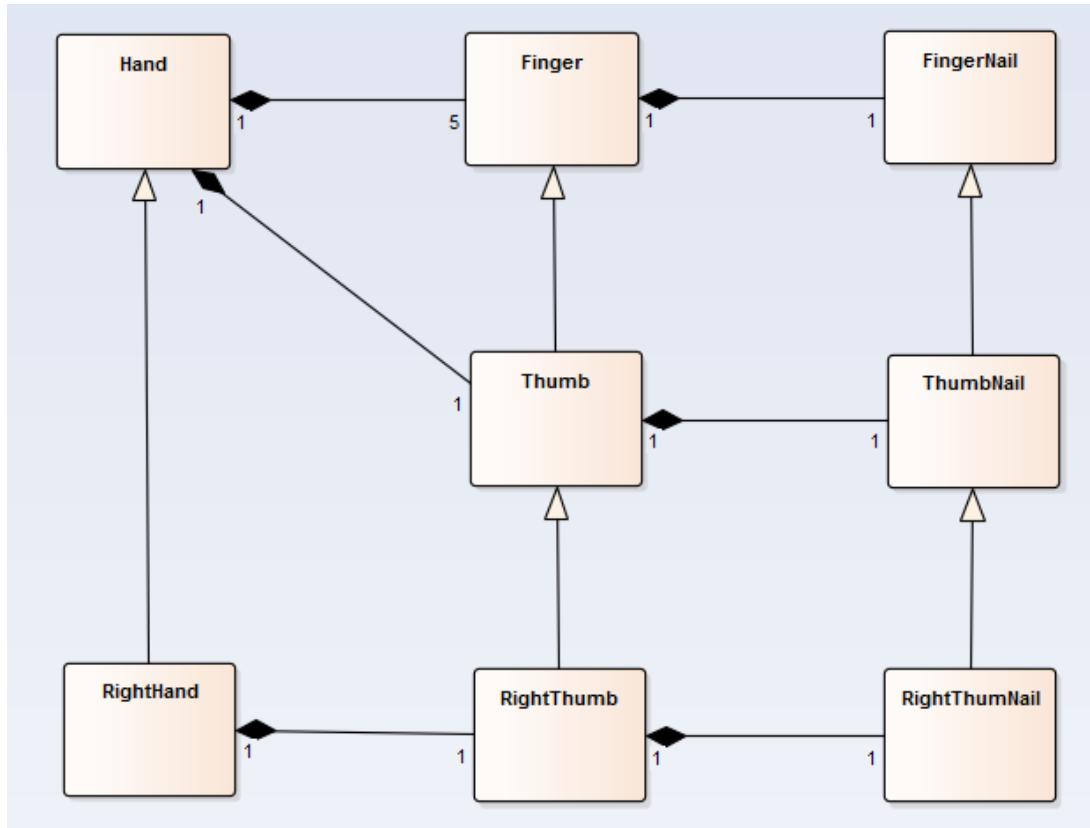


Partonomie (is-part-of- hierarchy)



# Wo? - Hierarchie und Abstraktion (6) – Kombination

Taxonomie (is-a hierarchy) und Partonomie (is-part-of- hierarchy)



Nach:

Quelle: Jansen, L.; Categories: The Top-Level Ontology; in: Munn, K.; Smith, B.; Applied Ontology – An Introduction, 2008

Komplexe Systeme bestehen aus mehreren Ebenen. Um eine Ebene darzustellen, sind die darin vorkommenden Elemente nur durch eine „ist\_benachbart\_mit“-Beziehung bzw. „thematic relation“ verbunden. Dabei sind nicht alle Elemente einer Ebene miteinander verbunden. Die Struktur der gesamten Ebene nennt man „**Heterarchie**“. Sie stellt eine „**Gleichordnung**“ der Elemente einer Ebene dar, da kein Element über- oder untergeordnet ist, sondern alle den gleichen Rang haben, bzw. derselben Ebene angehören. Eine Heterarchie besteht immer aus Elementen mit **verschiedenen Funktionen**. D.h. die Elemente einer Ebene haben sich differenziert. Die „Heterarchie“ stellt eine logische **AND-Beziehung** zwischen den Elementen einer Ebene dar. Innerhalb einer Ebene gibt es Elemente, die stärker miteinander in Beziehungen stehen. D.h. sie haben mehr Beziehungen untereinander als zu anderen Elementen der gleichen Ebene. Diese stärker miteinander in Beziehung stehenden Elemente bilden eine funktionale Einheit, die sich **Modul** nennt. Jedes Modul ist nach einem **Schema**, sprich besonderer Anordnung aufgebaut.

“Thematic relations refer to the link between concepts that occur together in time and space. Thematically related concepts play complementary roles in a given action or event ....”

Quelle: Jones, L.; Golonka, S.; Different influences on lexical priming for integrative, thematic, and taxonomic relations, 2012

„Thematic relations group ... concepts ... together by virtue of their participation in the same scenario or event.”

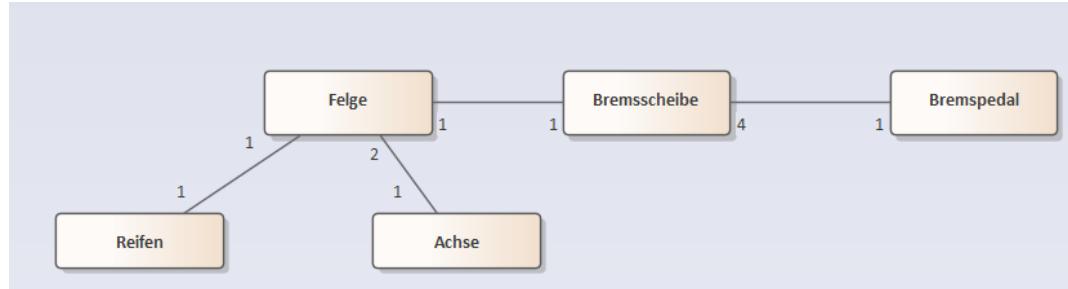
Quelle: Estes, Z. et al.; Thematic Thinking: The Apprehension and consequences of thematic relations, 2011

„... heterarchies, that is, networks of mutual influence without subordination. ... Their relation is one of reciprocity, or exchange among equals.“

Quelle: Heylighen, F., The Global Superorganism: an evolutionary-cybernetic model of the emerging network society, 2007

# Wo? - Heterarchie und Differenzierung (3)

Heterarchie (is-neighbour-of – Und-Beziehung)



Modularisierung / Differenzierung

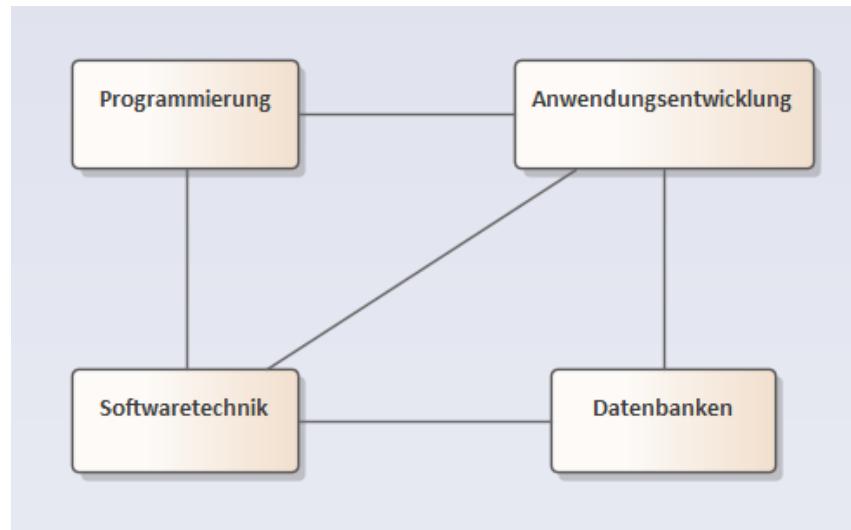
→ Viele Objekte

Funktionale Differenzierung (Modularisierung) der Elemente Bremsscheibe und Bremspedal des Bremssystems und Achse, Felge und Reifen des Fahrwerks.

D.h. in einer Heterarchie stehen auch gleichrangige Elemente unterschiedlicher Teildomänen/Modulen (Bremssystem, Fahrwerk) miteinander in benachbarter Beziehung (Felge und Bremsscheibe).

# Wo? - Heterarchie und Differenzierung (4)

Heterarchie (is-neighbour-of – Und-Beziehung)



Elemente der  
Vorlesungsebene des  
übergeordneten  
Elementes  
Wirtschaftsinformatik  
der  
Studiengangsebene

„If one considers ‘hierarchy’ in the context of organisational forms with sub- or super-ordination within different domains ... then the complementary term is ‘heterarchy’ can be defined as co-ordination.“

Quelle: von Goldammer, E. et al.; Heterarchy – Hierarchy – Two complementary categories of description, 2003

„A general-purpose definition [of heterarchy]... is the relation of elements to one another when they are unranked ... Heterarchy does not stand alone but is in a dialectical relationship with hierarchy (where elements are ranked). “

Quelle: Crumley, C.; Heterarchy, 2015

## Layering and partitioning

„The goal of system design is to manage complexity by dividing the system into smaller, manageable pieces. This can be done by a divide and conquer approach, where we recursively divide parts until they are simple enough to be handled by one person or one team. Applying this approach systematically leads to a hierarchical decomposition in which each subsystem, or layer, provides higher level services using services provided from lower level subsystems. Each layer can only depend on lower level layers and has no knowledge of the layers above it.,,

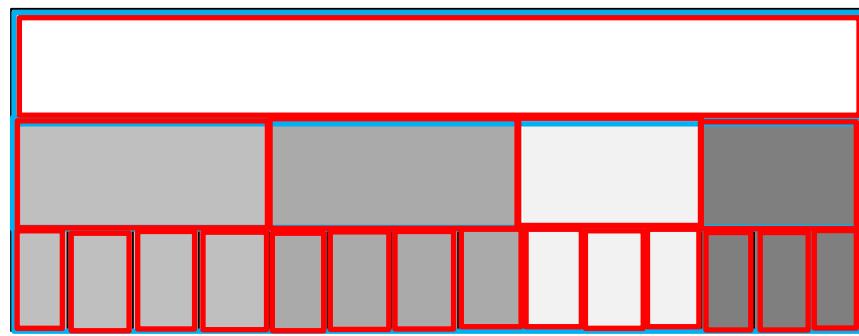
„Another approach to dealing with complexity is to partition the system into peer subsystems, each responsible for a different class of services.“

„In general, a subsystem decomposition is the result of both partitioning and layering.“

Quelle: Bruegge, B., Dutoit, A., Object-oriented Software Engineering, 3rd ed., 2010

Da die Entwicklung von Software komplex ist, weil funktionale und Qualitätsanforderungen sowohl von Fachexperten als auch von Informatikern berücksichtigt werden müssen, wird das Softwareprodukt in fachliche und technische Bereiche aufgeteilt. Beide Bereiche werden zusätzlich in logische Ebenen(Schichten) eingeteilt. Eine Ebene besteht aus mehreren Modulen, die miteinander in Beziehung stehen.

Skizzenhafte Darstellung (Seitenansicht)

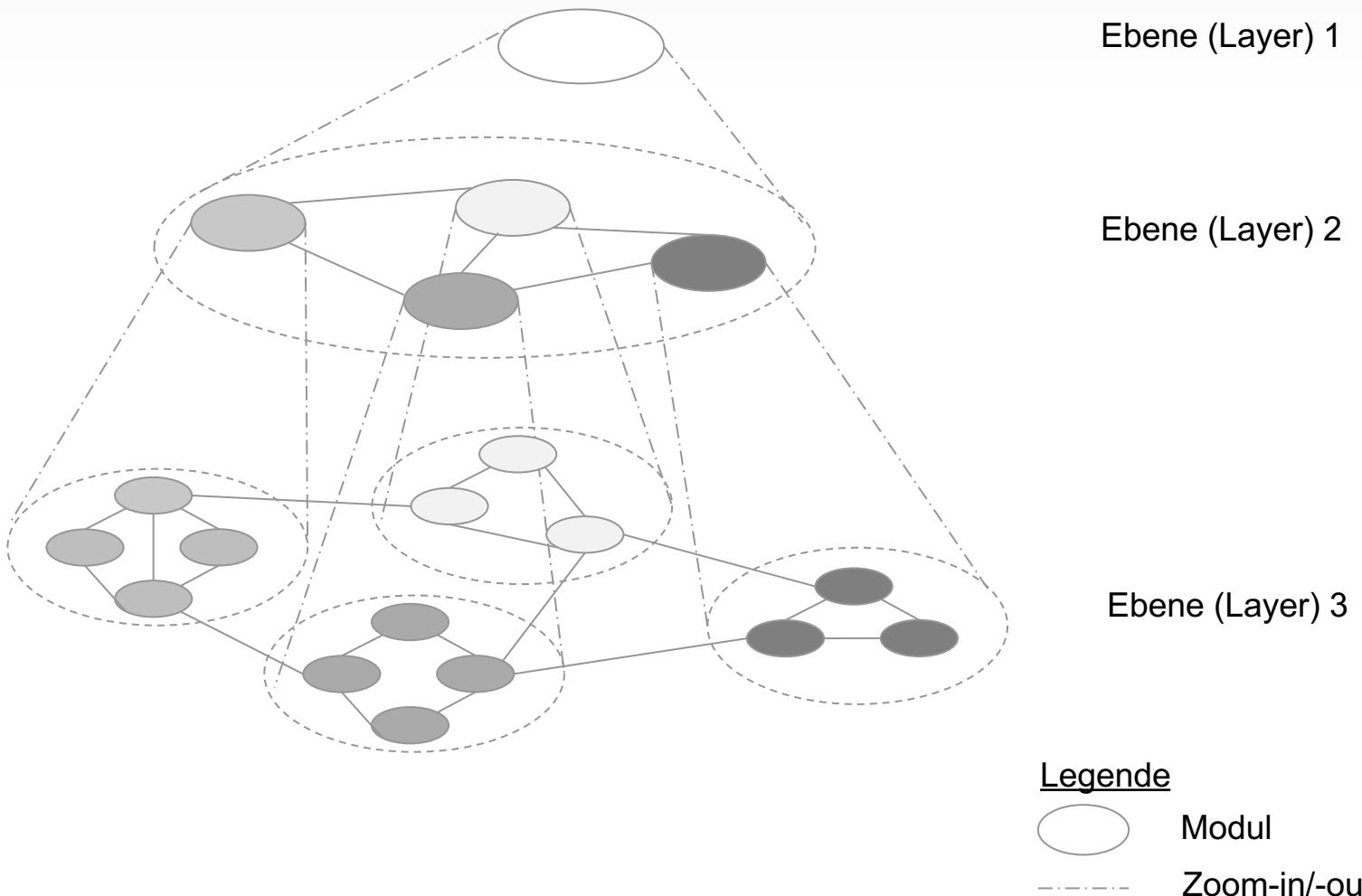


Achtung: keine Verbindungen zwischen den Modulen dargestellt!!!

## Legende

-  Ebene (Layer)
-  Modul (Module)

Skizzenhafte Darstellung (Draufsicht)



## Orientierung zwischen den Schichten (levels)

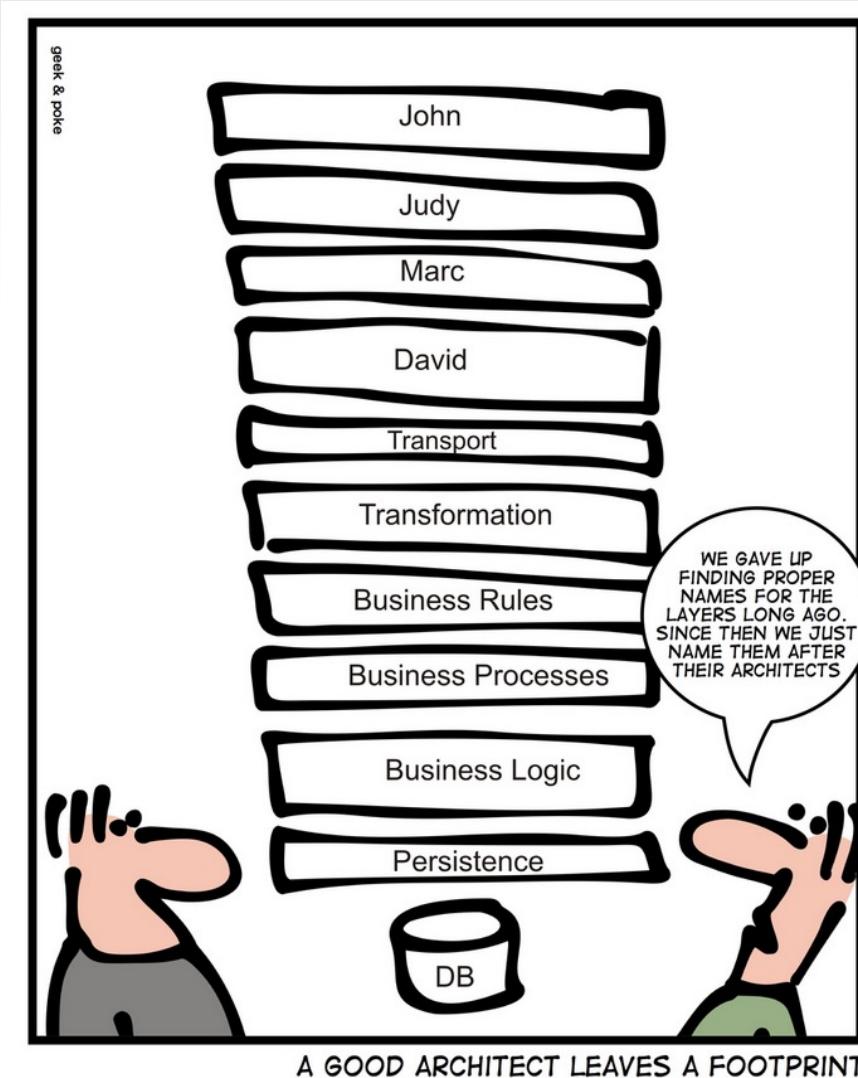
Zoom-in	Zoom-out
from blackbox to white box	from white box to blackbox
inner elements and relations visible	outer function visible
top-down	bottom-up
expand	collapse
one level down	one level up

Schichten sind relativ. D.h. es gibt keine absolut oberste und unterste Schicht.  
Wird eine Schicht (focal level) ausgewählt, dann gilt:

- Superordinate level
- Focal level
- Subordinate level

Auch blackbox und white box sind relativ. Das focal level ist die white box-Sicht des superordinate levels (blackbox), während das focal level die blackbox-Sicht des subordinate levels darstellt (white box).

# Wo? - Software – Bereiche und Ebenen/Schichten (4)



Quelle: geek & poke;

<https://static1.squarespace.com/static/518f5d62e4b075248d6a3f90/t/51e1b033e4b0c8784c9c6ec8/1434808002963/footprints2.jpg>

(Zugriff: 15.11.2016)

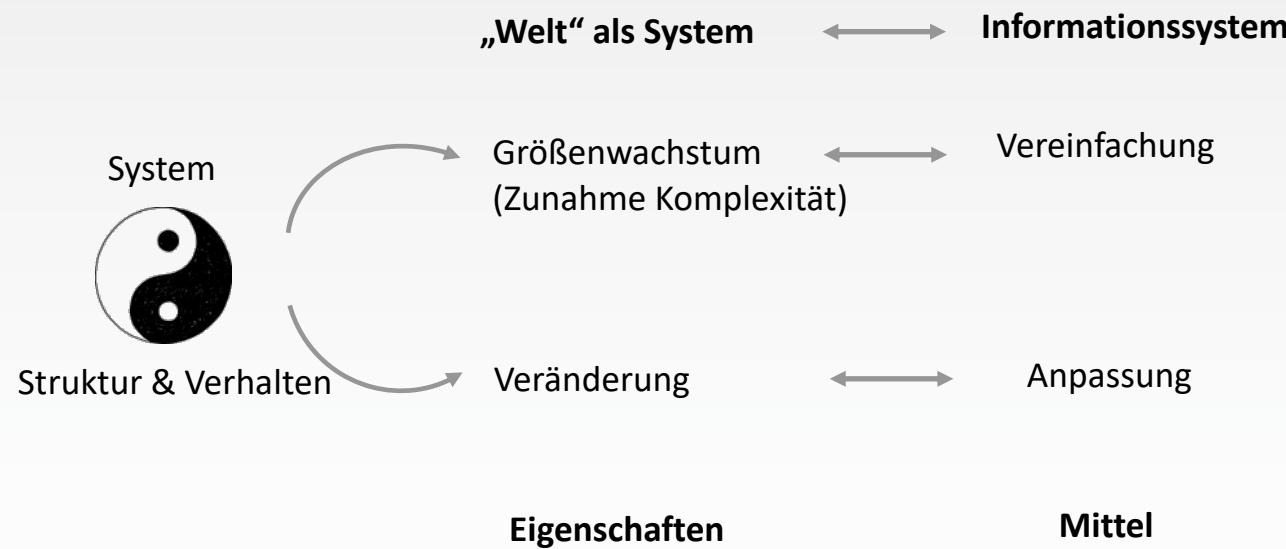
# Wie?

## „Komplexität ohne Ende“

Eine erste Annäherung an das Denken besteht in der Überlegung, dass Denken etwas mit Komplexitätsreduktion zu tun hat. Im Denken verarbeiten wir Rohdaten zu Informationen, indem wir zwischen Wesentlichen und Unwesentlichen unterscheiden. Dadurch können wir Muster in der Wirklichkeit erfassen. Diese Art von Komplexitätsreduktion ist eine Voraussetzung des Vermögens, uns mithilfe des Denkens in der Wirklichkeit zu orientieren.“

Gabriel, M.; Der Sinn des Denkens, 2018

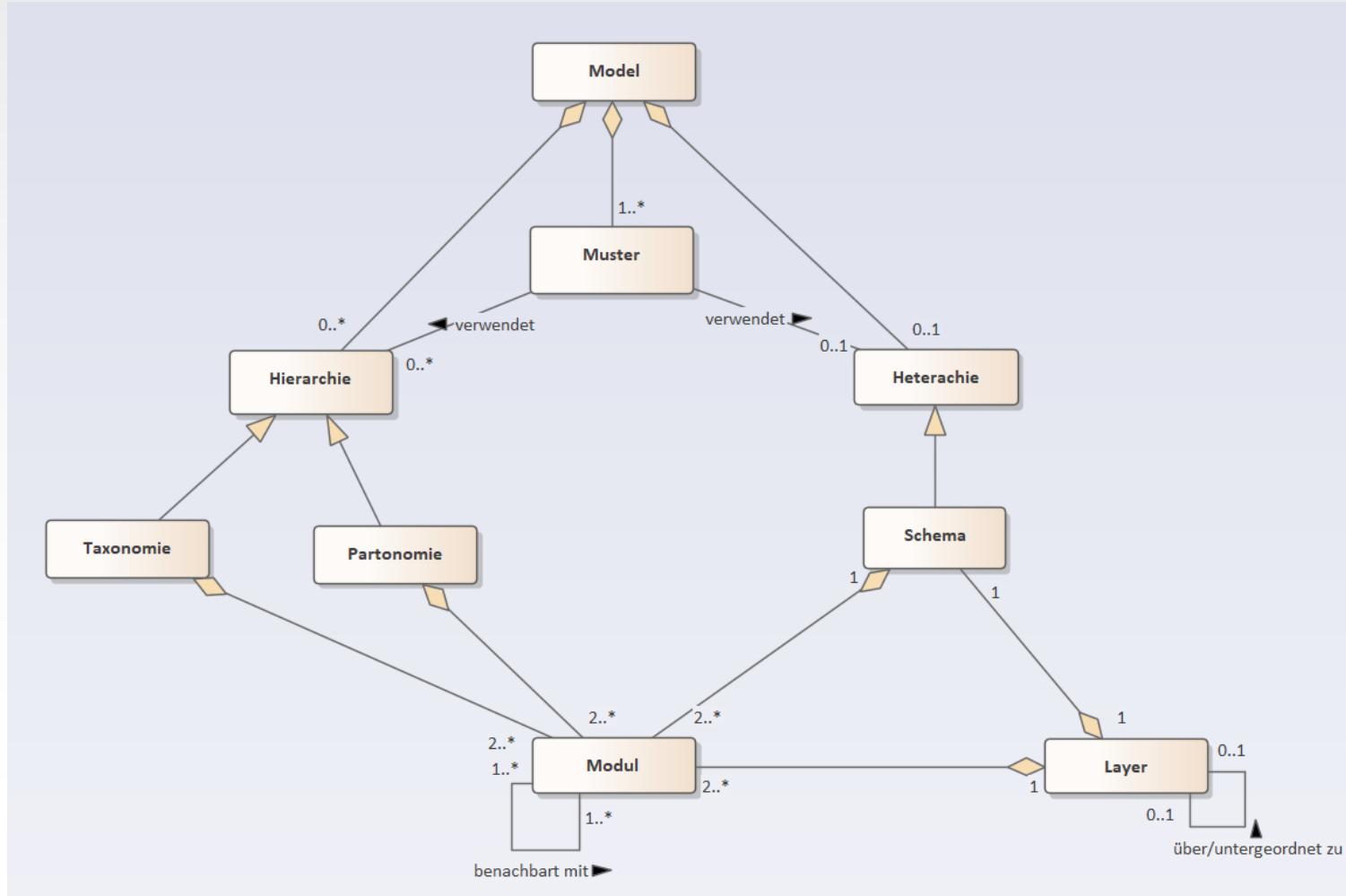
# Wie? - Systemkomplexität und allgemeine Lösungsansätze (2)



Vereinfachung → Anwendung von Methoden → strukturierte Vorgehensweisen

	Struktur (Was? / Ergebnis)	Verhalten (Wie? / Vorgehen)
Artefakte	Hierarchie	reduzieren, weglassen (abstrahieren)
	Partonomie	zusammenfügen (komponieren)
	Taxonomie	verallgemeinern (subsumieren)
	Heterarchie	differenzieren
	Schema	zusammenarbeiten (koordinieren, kollaborieren)
	Modul	horizontal unterteilen, abgrenzen (modularisieren)
	Layer	vertikal unterteilen, abgrenzen (verkleinern/vergrößern)
	Muster	ähnliches suchen (Gemeinsames im Verschiedenen entdecken)
	Model	abbilden (modellieren)
		Prinzipien

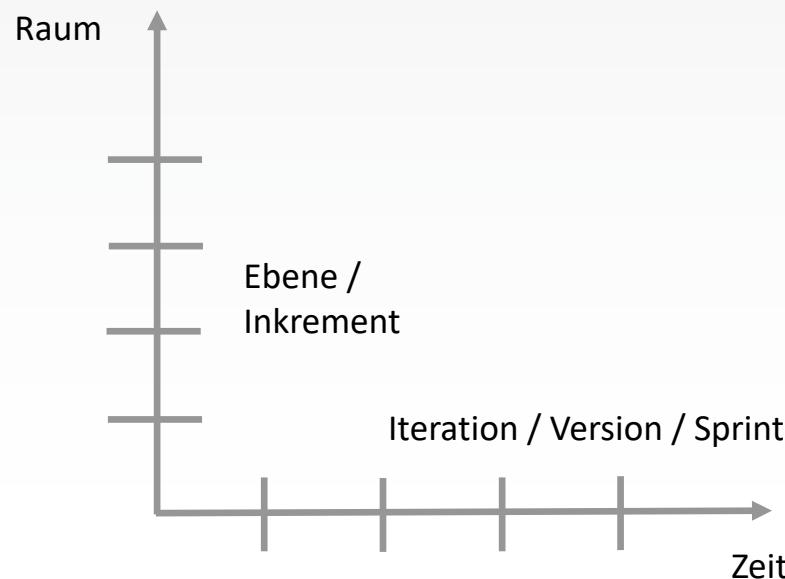
## Metamodell Artefakte des methodischen Vorgehens



Damit Menschen aufgrund ihrer geringen Speicher- und Verarbeitungskapazität die Umwelt erfassen und verändern können, haben sich über die Zeit allgemeine Prinzipien, Verfahren und Methoden herausgebildet um damit zurechtzukommen.

## Beispiel (Unterteilung)

In der Softwaretechnik werden die Dimensionen Raum bzw. Zeit in Ebenen / Inkrementen bzw. Iterationen / Versionen / Sprints unterteilt um somit kleinere und handhabbarer Einheiten zu erhalten.



„Grundsätze, die man seinem Handeln zugrunde legt.

- Allgemeingültig, abstrakt
- Sagen nichts darüber aus, wie man – bezogen auf ein konkretes Anwendungsgebiet – zu ihnen gelangt
- Beispiele
  - Abstraktion
  - Strukturierung
  - Hierarchisierung
  - Modularisierung“

Balzert, 2001

Eine Übersicht zu allgemeinen Prinzipien des objektorientierten Entwurfs finden Sie unter:

<http://www.iste.uni-stuttgart.de/se-old/links/links-se/entwurfsregeln-fuer-den-objektorientierten-entwurf/principles.html> (Zugriff: 09.12.2013)

„Unter **Abstraktion** versteht man die Verallgemeinerung, das Absehen vom Besonderen und Einzelnen, das Loslösen vom Dinglichen. Abstraktion ist das Gegenteil von Konkretisierung.“

„Eine **Struktur** ist ein Gefüge, das aus Teilen besteht, die wechselseitig voneinander abhängen.“

„Ein System besitzt eine **Hierarchie**, wenn seine Elemente nach einer Rangordnung angeordnet sind. Elemente gleicher Rangordnung stehen auf derselben Stufe, sie bilden eine Ebene bzw. Schicht der Hierarchie.“

Quelle. Balzert, H.; Lehrbuch der Softwaretechnik – Basiskonzepte und Requirements Engineering, 3. Aufl., 2009

„**Modularity**. The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on the other components.“

Quelle. IEEE Standard 610.12, 1990

## Beispiele:

„Prinzip:

Hierarchisierung

Methodische Vorgehensweise:

Zerlegung eines Problems in Teilprobleme, so dass eine Baumhierarchie entsteht

Prinzip:

Modularisierung

Methodische Vorgehensweise:

Entwicklung von Produkten, die nur über eine definierte Schnittstelle mit der Umwelt kommunizieren und sonst kontextunabhängig sind

Prinzip:

Strukturierung

Methodische Vorgehensweise:

Entwurf von Programmen, so dass nur Sequenz, Auswahl und Wiederholung vorkommen“

Balzert, 2001

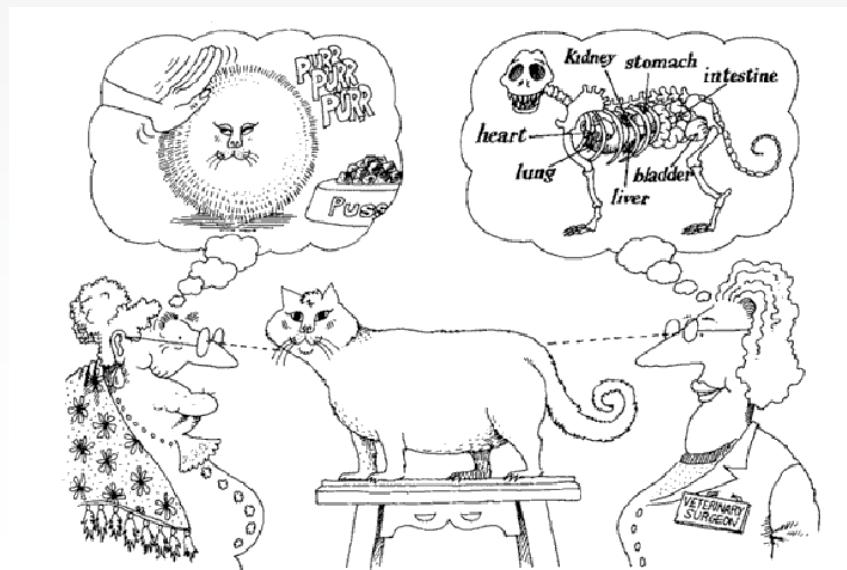
## Beispiele:

„Prinzip:

Abstraktion

Methodische Vorgehensweise:

Reduktion unwichtiger Eigenschaften bei gegebenen Einheiten aufgrund eines Ziels, so dass eine neue Einheit mit wesentlichen Eigenschaften entsteht.



Quelle: Booch, G. et al.; Object-oriented Analysis and Design with Applications, 3rd ed. Addison-Wesley, 2007, p. 45

## Prinzip Abstraktion

„Perhaps the most fundamental concept in systems modeling is abstraction, which concerns hiding unimportant details in order to focus on essential characteristics.“

Quelle: SEBok – Guide to the Systems Engineering Body of Knowledge  
[https://www.sebokwiki.org/wiki/System\\_Modeling\\_Concepts](https://www.sebokwiki.org/wiki/System_Modeling_Concepts); Zugriff: 11.11.2019

„...the most powerful abstractions are those that minimise the semantic gap between the units of analysis that are intuitively used to conceptualise the problem and the constructs present in the solution paradigm.“

Quelle: Jennings, N.; Wooldridge, M.; Agent-Oriented Software Engineering, 2000  
<http://icc.mpei.ru/documents/00000827.pdf>

Beispiel:

Das Konzept des Objektes ist sowohl innerhalb eines Wirklichkeitsausschnittes wie auch innerhalb der Objekt-Orientierung zu finden und verbindet daher als gemeinsame Vorstellung sowohl die fachliche Realität (z.B. Wirtschaft) mit der Informatik-Welt.

## Prinzip Modularisierung

„A network is *modular* if it can be partitioned into subsets of nodes (modules) that are densely connected internally but only sparsely connected to other subsets.“

Quelle: <http://rstb.royalsocietypublishing.org/content/367/1603/2704>; Zugriff: 17.05.2021

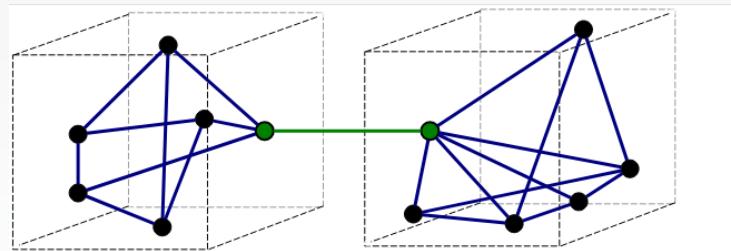
## Prinzip Modularisierung – coupling and cohesion

„Cohesion may be viewed as the glue that keeps the module together. ...

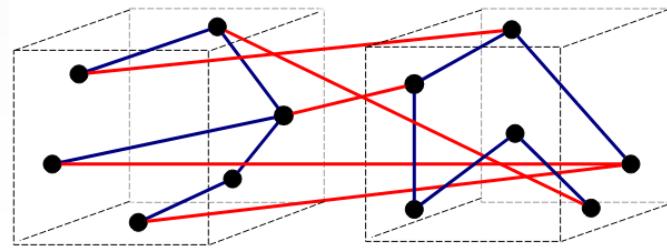
Coupling is a measure of the strength of the intermodule connections. ...

Coupling and cohesion are dual characteristics. If the various modules exhibit strong internal cohesion, the intermodule coupling tends to be minimal, and vice versa.“

Quelle: van Vliet, H., Software Engineering – Principles and Practices, 2007



a) Good (loose coupling, high cohesion)



b) Bad (high coupling, low cohesion)

Cohesion = intra-module relations  
Coupling = inter-module relations

Quelle: [https://en.wikipedia.org/wiki/Coupling\\_\(computer\\_programming\)](https://en.wikipedia.org/wiki/Coupling_(computer_programming)); Zugriff: 17.05.2021

## Prinzip Modularisierung – coupling and cohesion

„Coupling is the strength of dependencies between two subsystems. If two subsystems are loosely coupled, they are relatively independent, and thus, modifications to one of the subsystem will have little impact on the other. ... A desirable property of a subsystem decomposition is that subsystems are as loosely coupled as possible.“

„Coherence is the strength of dependencies within a subsystem. .... A desirable property of a subsystem decomposition is that it leads to subsystems with high coherence.“

„In general, there is a trade-off between coherence and coupling. We can always increase coherence by decomposing the system into smaller subsystems. However, this also increases coupling as the number of interfaces increases. A good heuristic is that developers can deal with  $7\pm 2$  concepts at any one level of abstraction. If there are more than 9 subsystems at any given level of abstraction or if there is a subsystem providing more than 9 services, you should consider a revision of the decomposition.“

„The goal of system design is to manage complexity by dividing the system into smaller, manageable pieces. This can be done by a divide and conquer approach, where we recursively divide parts until they are simple enough to be handled by one person or one team. Applying this approach systematically leads to a hierarchical decomposition in which each subsystem, or layer, provides higher level services using services provided from lower level subsystems. Each layer can only depend on lower level layers and has no knowledge of the layers above it..“

„Another approach to dealing with complexity is to partition the system into peer subsystems, each responsible for a different class of services.“

„In general, a subsystem decomposition is the result of both partitioning and layering.“

Quelle: Bruegge, B., Dutoit, A., Object-oriented Software Engineering, 3rd ed., 2010

## Prinzip Muster (Pattern)

Sind alle Smileys gleich? → Ja!

Ja → Grundaufbau / Struktur bzw. Organisation

Nein → Variationen der einzelnen Elemente

Common  
(Was? / Fest)



Variable  
(Wie? / Flexibel)



## Muster (Pattern)

„**Muster** bezeichnet allgemein eine statische Struktur, die durch ihr erneutes identisches Auftreten erkannt wurde. Es ist eine zur gleichförmigen Wiederholung (Reproduktion) bestimmte Denk-, Gestaltungs- oder Verhaltensweise bzw. einen entsprechenden Handlungsablauf.“

Quelle: <https://de.wikipedia.org/wiki/Muster>; Zugriff: 31.05.2017

„Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.“

Quelle: Alexander, C. et al.; A pattern language, 1977

„At the heart of science is a search for order. Science is grounded in the belief that the universe is neither arbitrary nor capricious, but follows patterns that can be discovered and understood.“

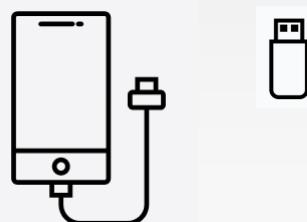
Quelle: System science handbook, 2021

## Muster (Pattern)

Hardware (USB-Interface)

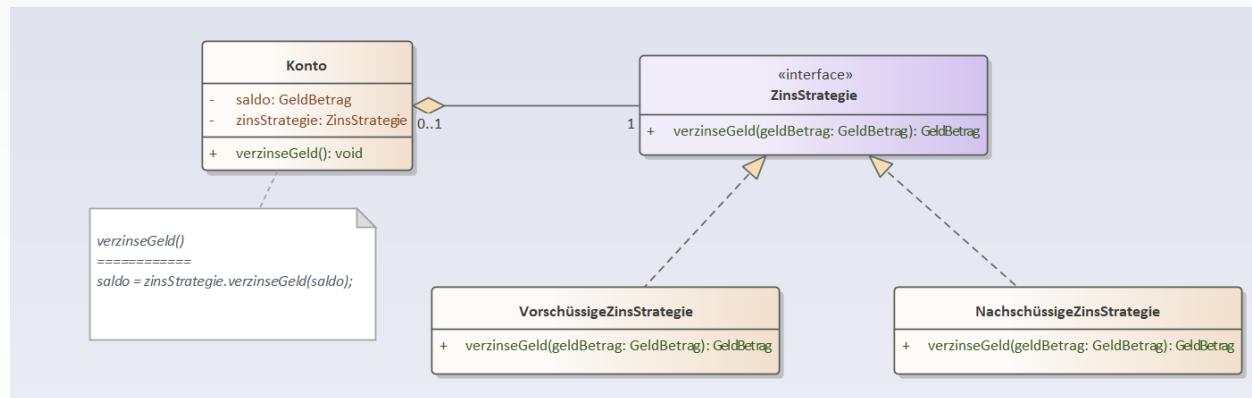


Common  
Was? / Fest



Variable  
Wie? / Flexible

Software (Strategy pattern)



Common  
Was? / Fest

Variable  
Wie? / Flexible

[https://de.freepik.com/freie-ikonen/usb-stick-beschrieben\\_742590.htm](https://de.freepik.com/freie-ikonen/usb-stick-beschrieben_742590.htm);  
<https://thenounproject.com/icon/cell-phone-with-usb-cable-2025259/>  
<https://de.depositphotos.com/57924313/stock-illustration-laptop-icon.html>

Zugriff: 06.05.2022

## Methoden

„Planmäßig angewandte, begründete Vorgehensweisen zur Erreichung von festgelegten Zielen“

Balzert, 2001

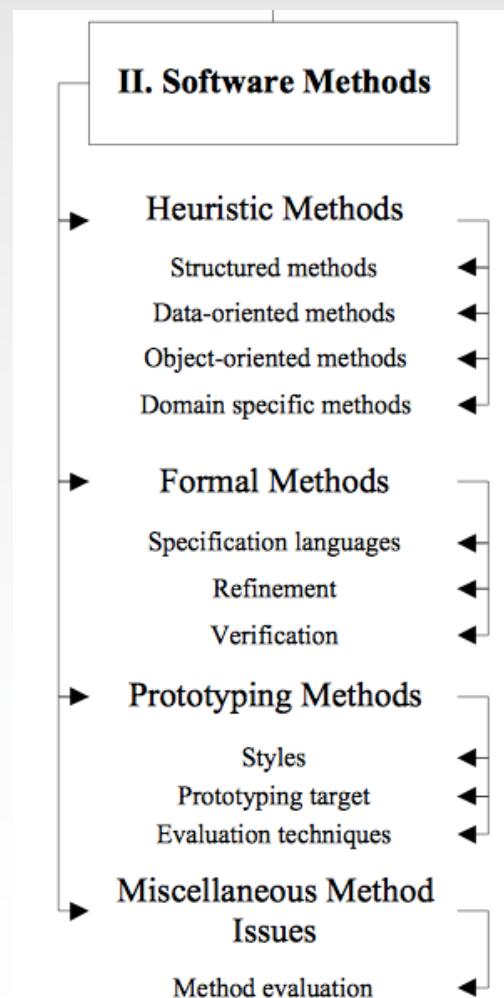
„A method is a disciplined procedure for generating a set of models that describe various aspects of a software system under development, using some well-defined notation.“

Booch, G. et al: Object-Oriented Analysis and Design with Applications, 3rd ed, 2007, S. 21

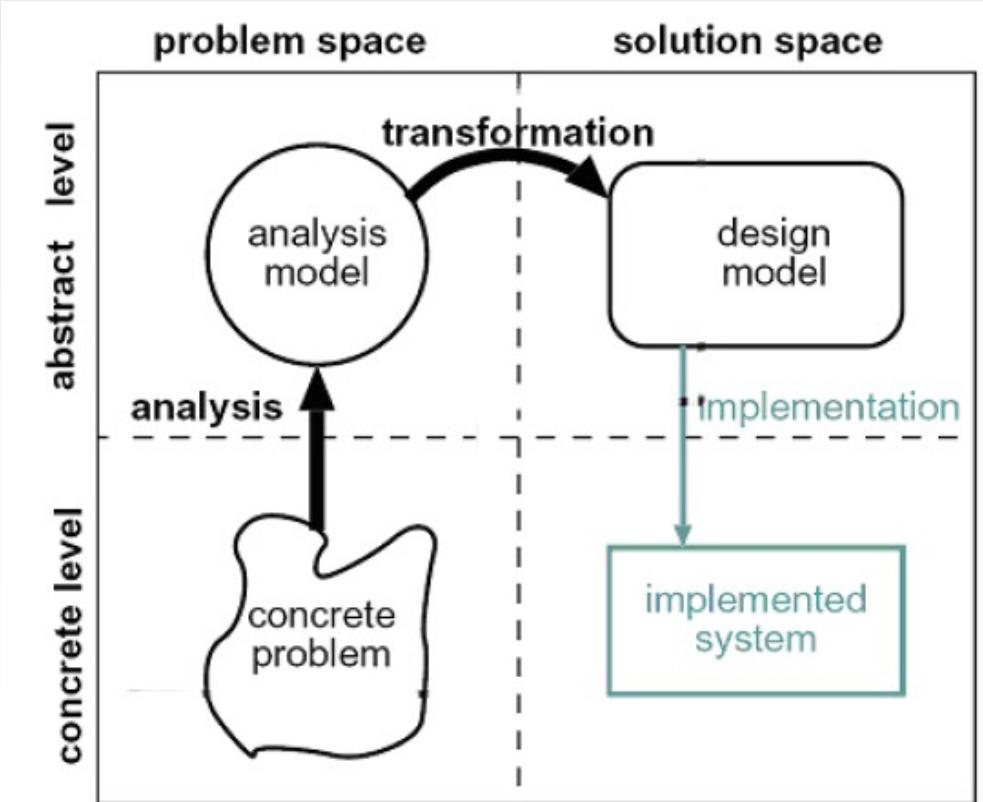
## Methodische Vorgehensweisen

- „Enthalten den Weg zu etwas hin
- Machen Prinzipien anwendbar
- Planmäßig:
  - Beim Einsatz einer methodischen Vorgehensweise wird **nicht** herumprobiert“

Balzert, 2001



Quelle: Bourque, P., Fairley, R.; SWEBOK V3.0 – Guide to the Software Engineering Body of Knowledge, IEEE, 2004



Quelle: Hahsler, M.; Analyse Patterns im Softwareentwicklungsprozeß, Wien 2001

„System analysis is the analysis of a problem that a firm tries to solve with an information system. It consists of defining the problem, identifying its causes, specifying the solution, and identifying the information requirements that must be met by a system solution.“

Quelle: Laudon, K.; Laudon, J.; Management Information Systems: Managing the Digital Firm; 2017

Unter **Systemanalyse** (im engeren Sinn) versteht man die **Zerlegung** eines Systems nach bestimmten Aspekten.

In der Systemanalyse werden die (räumlichen) **Strukturen** und das (zeitliche) **Verhalten** des Systems untersucht. Zusätzlich wird der **Zustand** des Systems, also die Änderung von Strukturen durch das Verhalten, betrachtet.

Als Systemanalyse im weiteren Sinn bezeichnet man zusätzlich noch die Aspekte „**Systemdesign**“ und „**Systemforschung**“.

Für die anwendungsorientierte Softwareentwicklung ist die Systemanalyse und das System-design von entscheidender Bedeutung.

Die Softwareanalyse kann physische System (z.B. Fahrrad) oder virtuelle Systeme (Finanzsystem) untersuchen. In der Regel betrachtet die Systemanalyse im Bereich Wirtschaftsinformatik virtuelle Systeme.

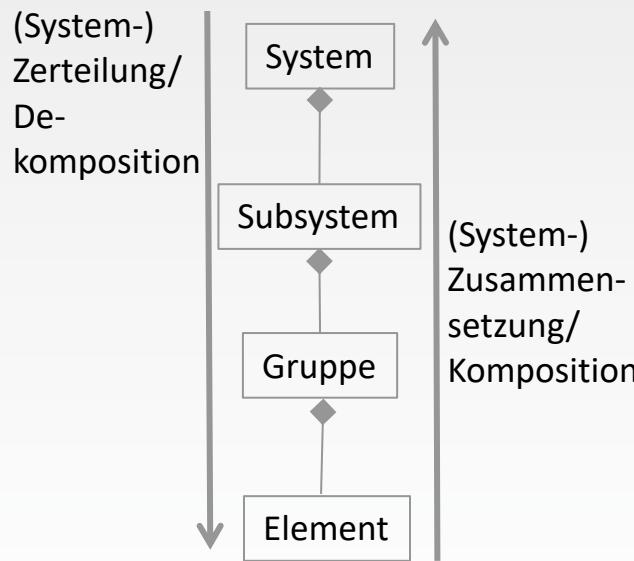
Virtuelle Systeme können dahingehend untersucht werden, dass ein logischer Raum mit 3 Dimensionen angenommen wird. Die eine Dimension betrachtet die Zusammensetzung/Zerteilung („ist\_Teil\_von“-Beziehung zwischen Einheiten“, ein andere Dimension zeigt die Verallgemeinerung/Konkretisierung von Begriffen („ist\_ein“-Beziehung) auf und eine weitere Dimension beschäftigt sich mit den Verbindungen von Einheiten und Begriffen auf gleicher Ebene („ist\_verbunden\_mit“-Beziehung) .

In der **Systemanalyse** wird durch eine „**top-down**“ Vorgehensweise das System zerlegt (eine gröbere Sicht auf das System wird schrittweise immer detailreicher ausgearbeitet). Bei der **Systemsynthese** wird durch eine „**bottom-up**“ Vorgehensweise das System zusammengesetzt (eine detailreichere Sicht auf das System wird schrittweise immer größer ausgearbeitet).

# Wie? - Vorgehensweise (4)

## räumliche Dimension (zwischen und auf gleicher Ebene)

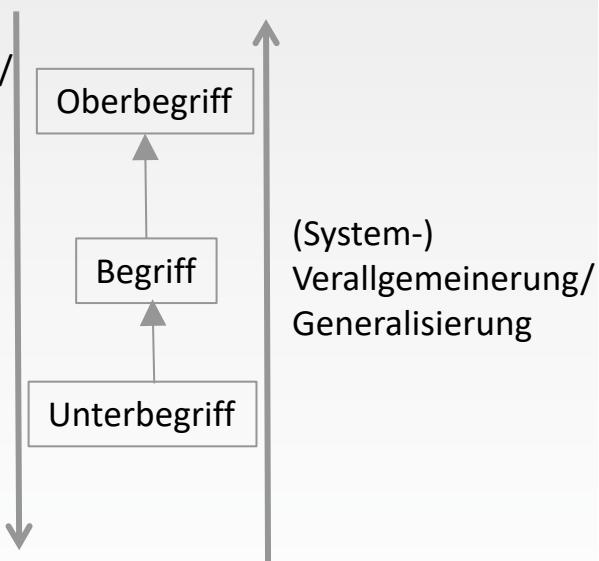
### Partonomie



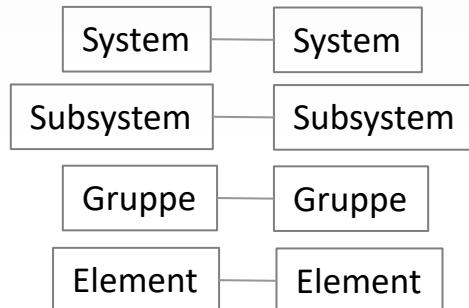
### Hierarchie



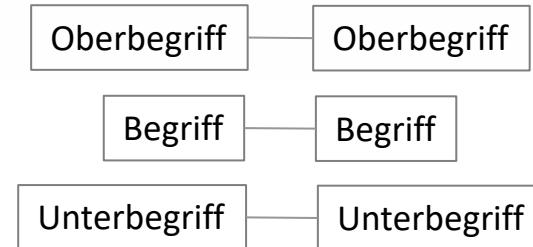
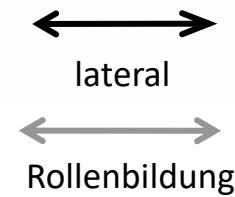
### Taxonomie



### Schema

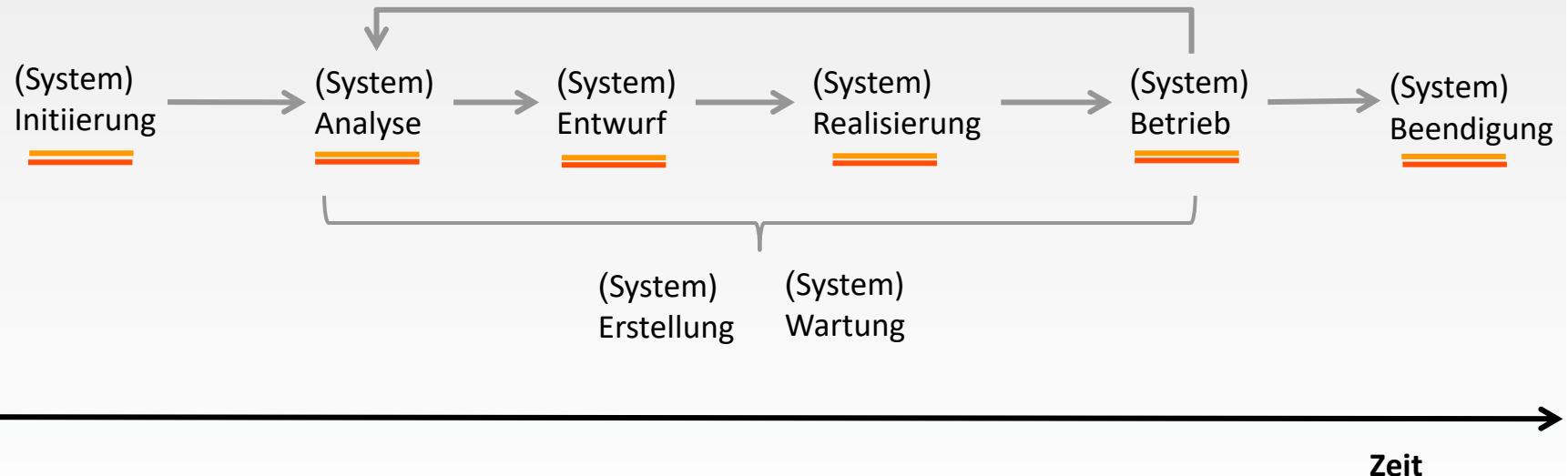


### Heterarchie



## zeitliche Dimension (vom Fachlichen zum Technischen)

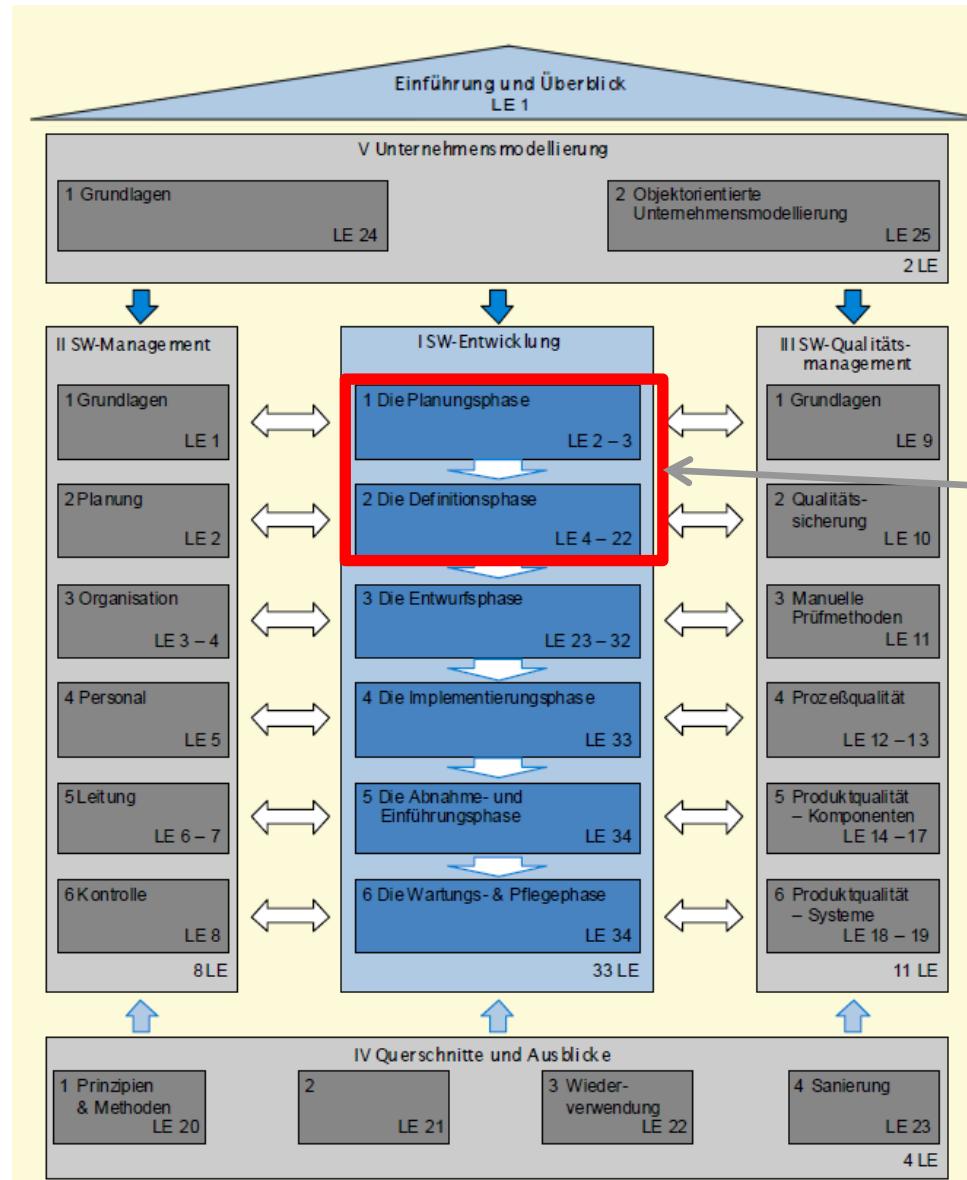
### Software Development Life Cycle (SDLC)



#### Legende

 Querschnittsaufgaben (z.B. Testen, Dokumentation)

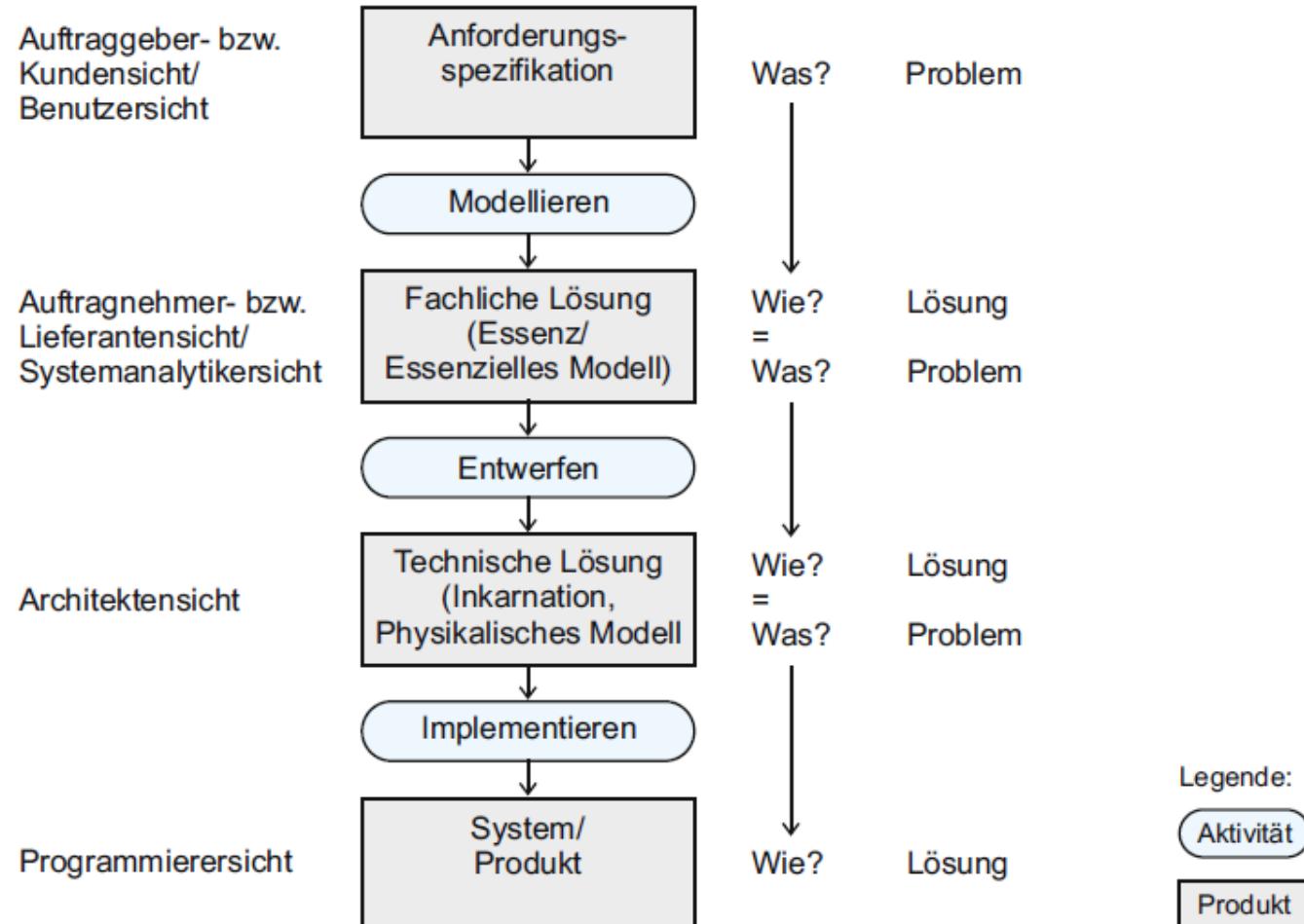
# Wie? - Vorgehensweise (6)



Quelle: Balzert, H.; Lehrbuch der Softwaretechnik, 2001

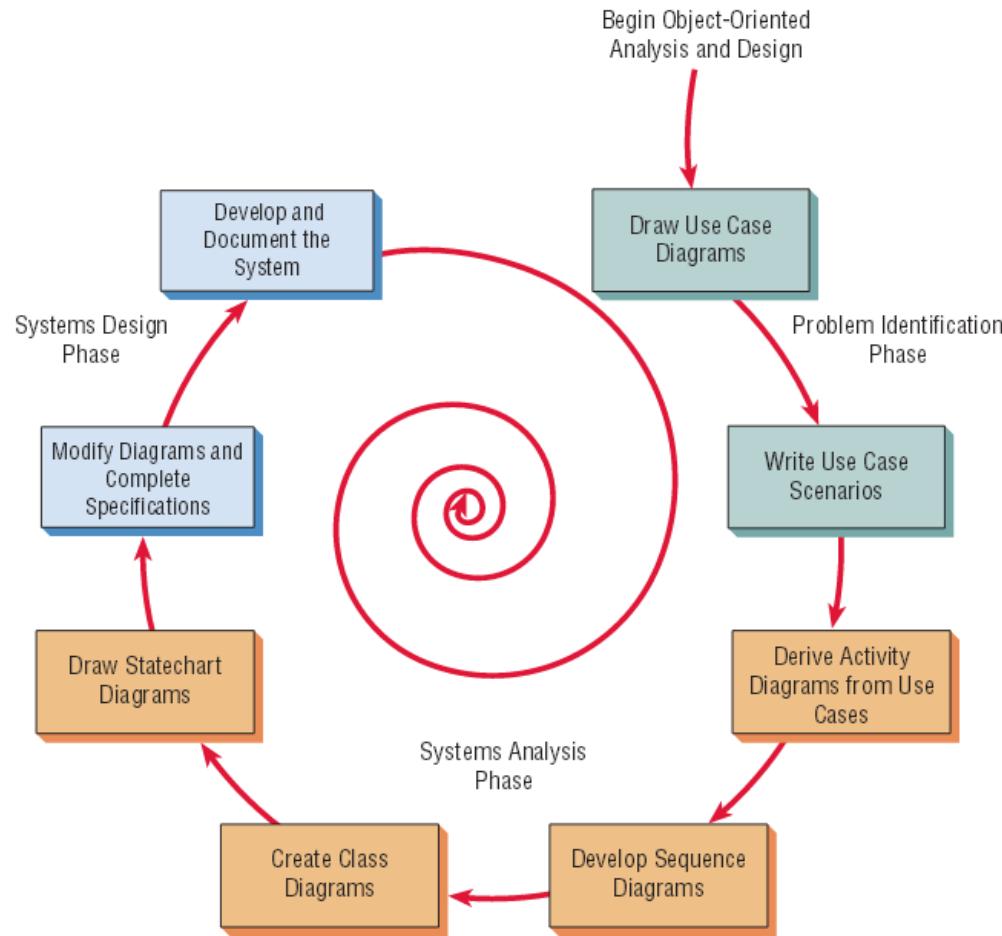
# Wie? - Vorgehensweise (7)

## Vorgehensschritte und Artefakte



Quelle: Balzert, H.; 2009

## Object-oriented analysis



Quelle: Kendall, K., Kendall, J.; Systems analysis and design, 8th ed., 2011

# Wer?

Die Frage Wer? fragt nach dem aktiven, also handelnden, Subjekt in einer Beziehung. Als Subjekte kommen generell Menschen und Maschinen in Betracht.

Menschen oder Maschinen, die aktiv eine Beziehung innerhalb oder außerhalb eines Informationssystem eingehen, heißen Aktoren (actor). Aktiv ist ein Aktor, wenn er selbstständig Aktionen auslösen kann.

Generell sind folgende Kombinationen der Interaktion zwischen Aktoren denkbar:

- Mensch – Mensch
- Mensch – Maschine
- Maschine – Maschine

Es lassen sich aus den Basisinteraktionskombinationen Interaktionsketten bzw. –netze bilden. Z.B.:

- Mensch – Maschine – Mensch
- Maschine – Maschine – Mensch

Für die Akteure außerhalb und innerhalb eines Systems haben sich aufgrund der Aufgabenteilung (Komplexitätsbewältigung) funktionale Rollen herausgebildet.

Beispiele sind:

- Mensch – Mensch (AuftraggeberIn – AuftragnehmerIn)
- Mensch – Maschine (BedienerIn – Bediengerät)
- Maschine – Maschine (Client – Server)

## Personen und Rollen



Quelle: <https://www.goodtherapy.org/blog/blog/wp-content/uploads/2013/01/woman-roles.jpg>; Zugriff: 21.01.2015

# Wer? - Rolle (2)



Sportler (Rolle)



Angestellter (Rolle)



Person (Kern)



Bauarbeiter (Rolle)



Gast (Rolle)

Quelle: (Zugriff: 22.01.2018)  
[http://de.simpsons.wikia.com/wiki/Homer\\_Simpson](http://de.simpsons.wikia.com/wiki/Homer_Simpson)  
<http://www.businesswire.com/news/home/20070716006230/en/Doh-I-Homer-Simpson-Publicity-Stop-NBCs-Tonight>  
<https://www.mirror.co.uk/sport/other-sports/american-sports/cartoon-hero-homer-simpson-inducted-9911607>  
<https://www.giga.de/filme/die-simpsons/specials/nach-dem-tod-von-homers-synchronsprecher-wie-geht-es-mit-den-simpsons-weiter/>

# Wer? - Rolle (3)

Ein Objektkern schlüpft – meistens nacheinander – in unterschiedliche Objektrollen. Der Objektkern sowie die Objektrolle werden durch Eigenschaften (Attribute) und Verhalten (Methoden) genau beschrieben.



Person (Kern)

## Eigenschaften (Attribute)

- Geschlecht
- Alter

## Verhalten (Methoden)

- atmet
- beobachtet



Sportler (Rolle)



Angestellter (Rolle)



Gast (Rolle)



Bauarbeiter (Rolle)

## Eigenschaften (Attribute)

- Laufkleidung

## Verhalten (Methoden)

- läuft

## Eigenschaften (Attribute)

- Arbeitskleidung

## Verhalten (Methoden)

- denkt

## Eigenschaften (Attribute)

- Festkleidung

## Verhalten (Methoden)

- beantwortet Fragen

## Eigenschaften (Attribute)

- Arbeitskleidung

## Verhalten (Methoden)

- arbeitet

Quelle: (Zugriff: 22.01.2018)

[http://de.simpsons.wikia.com/wiki/Homer\\_Simpson](http://de.simpsons.wikia.com/wiki/Homer_Simpson)

<https://www.businesswire.com/news/home/20070716006230/en/Doh!-Homer-Simpson-Publicity-Stop-NBCs-Tonight>

<https://www.mirror.co.uk/sport/other-sports/american-sports/cartoon-hero-homer-simpson-inducted-9911607>

<http://www.giga.de/filme/die-simpsons/specials/nach-dem-tod-von-homers-synchronsprecher-wie-geht-es-mit-den-simpsons-weiter/>

# Wer? - Rolle (4)

Objektkern

+

Objektrolle

=

Beziehungsobjekt



Person (Kern)

Gast (Rolle)

Person (Kern) +  
Gast (Rolle)

Person (Kern) +  
Gastgeber (Rolle)

Eigenschaften (Attribute)

- Geschlecht
- Alter

Eigenschaften (Attribute)

- Festkleidung

Eigenschaften (Attribute)

- Geschlecht
- Alter
- Festkleidung

Eigenschaften (Attribute)

- Geschlecht
- Alter
- Festkleidung

Verhalten (Methoden)

- atmet
- beobachtet

Verhalten (Methoden)

- beantwortet Fragen

Verhalten (Methoden)

- atmet
- beobachtet
- beantwortet Fragen

Verhalten (Methoden)

- atmet
- beobachtet
- stellt Fragen

Quelle: (Zugriff: 22.01.2018)

[http://de.simpsons.wikia.com/wiki/Homer\\_Simpson](http://de.simpsons.wikia.com/wiki/Homer_Simpson)

<https://www.businesswire.com/news/home/20070716006230/en/Doh!-Homer-Simpson-Publicity-Stop-NBCs-Tonight>

<http://archive.indianexpress.com/news/homer-simpson-a-true-catholic-vatican-newspaper/699067/>

Vorsicht! – Verwechslung bei „ist\_ein“-Beziehung

„ist\_ein\_Rolle\_von“ (Kern-Rolle)  $\neq$  „ist\_ein(e\_Art\_von)“ (Generalisierung-Spezialisierung)

Objektebene		
Beispiel	Beziehungsart	Bindungsstärke
Jörg ist ein Mensch	Klassifizierung - Instanzierung	Sehr fest
Jörg ist ein Freund (von Lisa)	Objekt - Rolle	Sehr lose
Typebene		
Beispiel	Beziehungsart	Bindungsstärke
Ein Mensch ist ein Säugetier	Generalisierung - Spezialisierung	Sehr fest
Ein Mensch ist ein Studierender (eines Studiengangs)	Kern - Rolle	Sehr lose

Als Rolle kann eine Beschreibung von **Eigenschaften** und **Verhalten unter einem Namen** angesehen werden, die ein Element **in einer Beziehung** zu einem anderen einnimmt.

Ein **Aktor** (Mensch / Maschine) kann mehrere (Funktions-)Rollen annehmen und eine (Funktions-)Rolle kann von mehreren Akteuren ausgeübt werden. Allerdings kann zu einem Zeitpunkt nur eine Rolle von einem Akteuren angenommen werden.

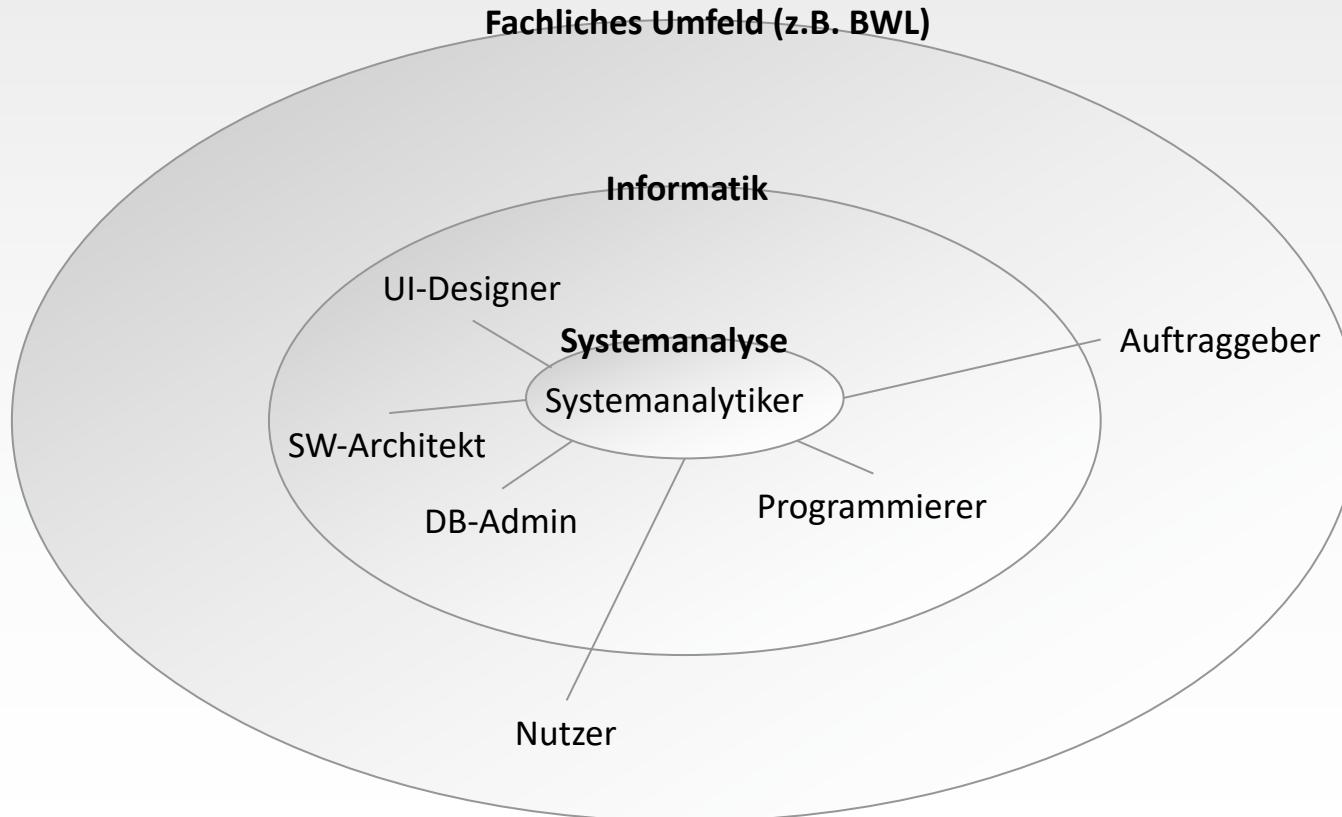
Rollen, die im Bereich der Softwareentwicklung von Personen eingenommen werden, sind z.B. Systemanalytiker, Softwarearchitekt, Programmierer, Datenbankadministrator, Softwaretester, User-Interface-Designer.

Rollen, die von Artefakte im Bereich Softwareentwicklung eingenommen werden, sind z.B. Client, Server, Peer, Broker.

Um eine Rolle entsprechend zu erfüllen, sind für diese Rolle **spezifische Aufgaben** definiert, die bei Bedarf durch **Techniken (Werkzeuge, Methoden), Sprachen und Modelle** unterstützt werden. Dies ist dadurch begründet, dass das entsprechende Subjekt nur einen **Teilausschnitt** der zu betrachtenden „Welt“ untersuchen und gestalten soll (Rollendefinition).

Nur durch eine **Arbeitsteilung** in Form von spezialisierten Rollen und deren Zusammenwirken in einem Projekt kann die **Komplexität** der „Welt“ bewerkstelligt werden.

## Rollen / Stakeholder im Softwareentwicklungsprojekt – Schwerpunkt Systemanalyse



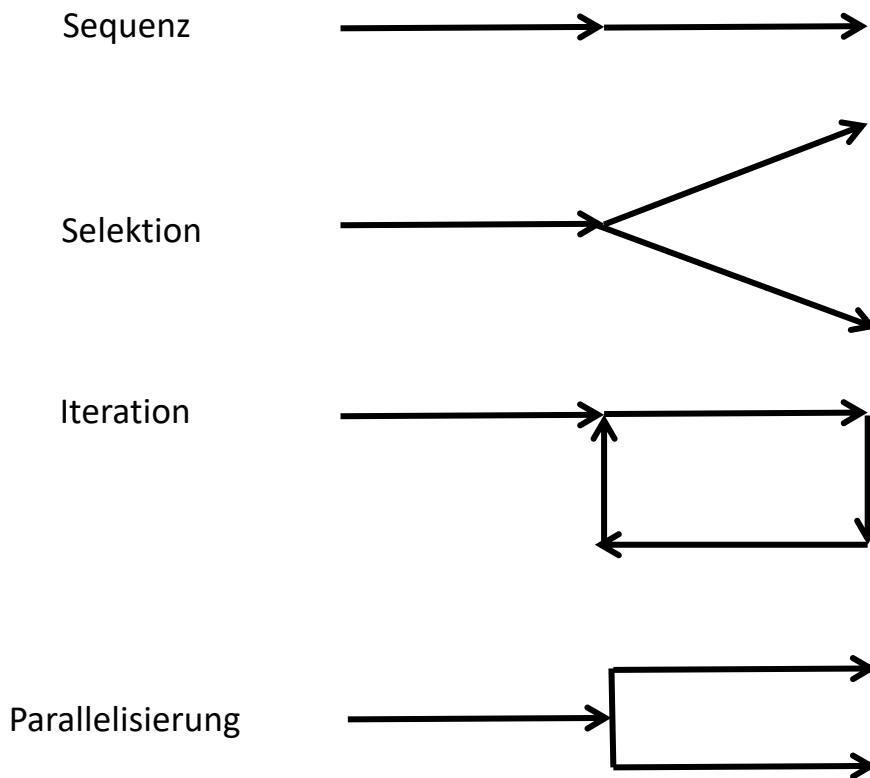
# Wer? - Rolle (8)

Rolle / Stakeholder	Beispiele für Elemente der Rollensicht	Name des Modells	Sprache
Nutzer(in) / Auftraggeber(in)	<ul style="list-style-type: none"> <li>• Kosten</li> <li>• Zeit</li> <li>• Qualität</li> <li>• Aufgabe / Zweck</li> <li>• Fachbegriffe</li> </ul>	Fachmodell Finanzierungsmodell	Textuelle Fachsprache (z.B. BWL)
Systemanalytiker(in)	<ul style="list-style-type: none"> <li>• Anforderung</li> <li>• Fachkonzepte</li> </ul>	Analysemodell	Graphische Modellierungssprache (z.B. BPMN, UML)
SW-Architekt(in)	<ul style="list-style-type: none"> <li>• Muster</li> </ul>	Architekturmodell	Graphische Entwurfssprache (z.B. UML)
DB-Designer	<ul style="list-style-type: none"> <li>• Entität</li> <li>• Relation</li> </ul>	Entity-Relationship-Model (ERM)	Graphische Designsprache
UI-Designer(in)	<ul style="list-style-type: none"> <li>• Widget</li> <li>• Layout</li> </ul>	UI-Model (z.B. Wireframe)	Graphische Designsprache
Programmierer(in)	<ul style="list-style-type: none"> <li>• Klasse</li> <li>• Programm</li> <li>• Datenstruktur</li> <li>• Algorithmus</li> </ul>	Programmiermodell	Textuelle Programmiersprache (z.B. Java)

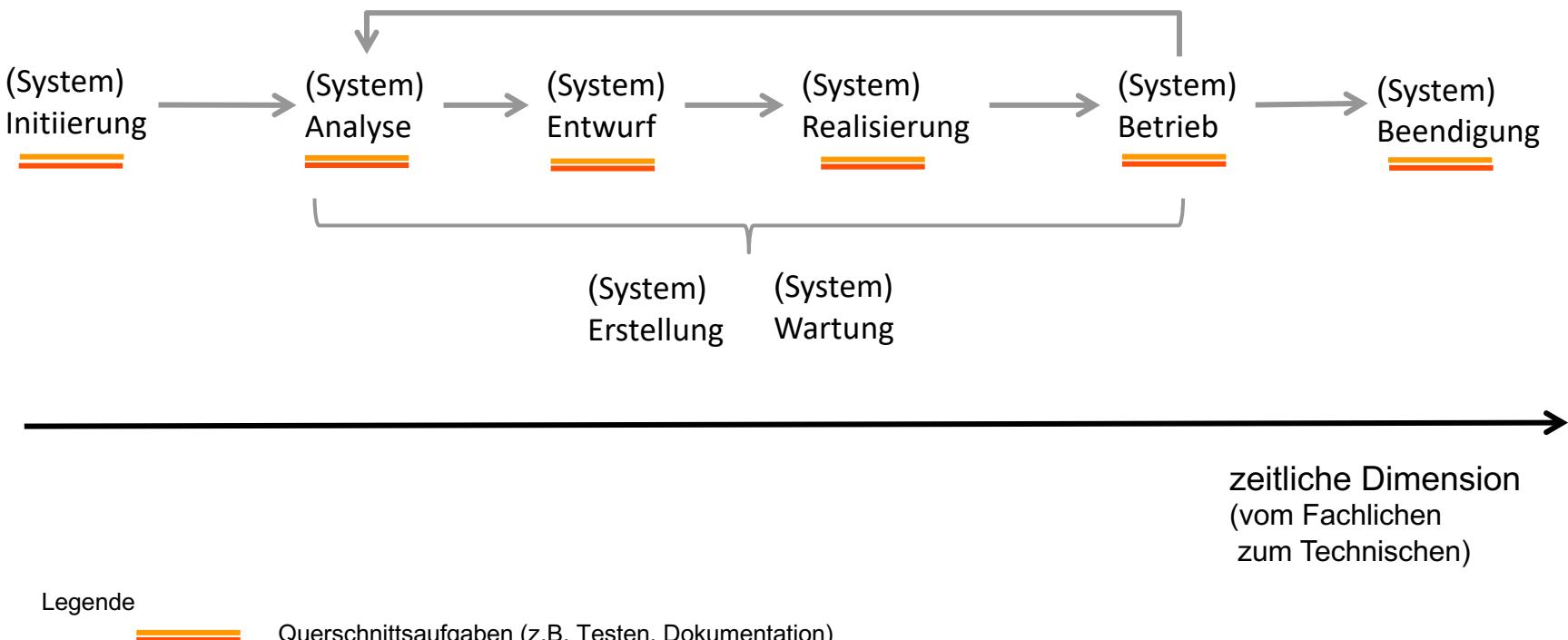
# Wann?

Zur Gliederung der Dimension „Zeit“ werden in der Informatik die 4 Kontrollstrukturen für die Abfolge (Sequenz), Wiederholung (Iteration), Verzweigung (Selektion) und Nebenläufigkeit (Parallelisierung) von Aktionen verwendet. Die Kombination der 4 Kontrollstrukturen ergibt das Verhalten des Softwaresystems so wie es auch in der Realität beobachtet, modelliert und implementiert wurde.

Alle Verhaltensmodelle verschiedener Sprachen/Notationen enthalten explizit die 4 Kontrollstrukturen (z.B. UML, BPMN, imperative Programmiersprachen (z.B. Java)).



Da die Entwicklung von Software komplex ist, weil neben den funktionalen und Qualitätsanforderungen der Fachexperten auch technische Erfordernisse berücksichtigt werden müssen wie z.B. Grafische Oberflächen, Datenbankmanagementsysteme, wird die zeitliche Entwicklung von Software in logische Phasen eingeteilt. Jede Phase hat spezielle Aufgaben und ihr zugewiesene Stakeholder/Rollen sowie Ein- und Ausgangsartefakte.  
Die Verkettung der einzelnen logischen Phasen des Software development life cycle (SDLC) wird Vorgehensmodell / Prozessmodell genannt.



Jenkins, G., The systems approach , 1972 “described 4 phases in the systems engineering approach:

1. Systems analysis
2. Systems design
3. Implementation
4. Operation“

in: Hitchins, D., Putting systems to work, 1993

# Wann? - Vorgehensmodell (3)

Rolle	Modell	Sprache	Artefakte	Begriffe	W-Fragen	Phase
Domänen-spezialist(in)	Domänen-modell	Fachsprache	Beschreibungen	Konto, Zahlung, Kosten	Was?	Analyse
System-analytiker(in)	Domänen-modell	UML, BPMN	Lasten-/Pflichtenheft, Product Backlog	Use Case (plan-basiert) User Story (agil)	Was?	Analyse
Software-architekt(in)	Architektur-modell	UML, ADL	Komponenten- und Verteilungsdiagramm	Komponente & Konnektor, Schnittstelle, Muster	Wie?	Entwurf
(G)UI/Web-Designer(in)	(G)UI-Modell	Grafische Skizzen	Wireframe, Dialog	UX, Widget, Dialog, Landing page	Wie?	Entwurf
DB-Designer(in)	DB-Design-modell	ERM	DB-Designmodell	Entität, Relation, Knoten, Kante	Wie?	Entwurf
Anwendungs-programmierer(in)	Programmier-modell	Java, Python	Quellcode	Klasse, Bibliothek	Womit?	Implementierung
(G)UI/Web-Developer(in)	Web-Client-Modell	HTML, CSS, Javascript	Single-Page-Application	Web-Client/Server	Womit?	Implementierung
DB-Realisier(in)	DB-Realisierungs-modell	SQL, GQL	DB-Schema	Tabelle, Graph, Fremdschlüssel	Womit?	Implementierung
Cloud-Admin	Verteilungs- und Virtualisierungsmodell	Cloud-Konfigurations-sprache	Cloud-Betriebskonzept	Cloud, Microservice, Container	Wo?	Betrieb
DB-Admin	Verteilungs-modell	DB-Konfigurations-sprache	DB-Betriebskonzept	Datenbank, Index, Sharding, Replikation	Wo?	Betrieb

„Jede Software-Entwicklung soll in einem festgelegten organisatorischen Rahmen erfolgen. Ein **Prozess-Modell** – auch **Vorgehensmodell** genannt – beschreibt einen solchen Rahmen. In ihm wird festgelegt, welche Aktivitäten in welcher Reihenfolge von welchen [Rollen] erledigt werden und welche Ergebnisse ... dabei entstehen und wie diese in der Qualitätssicherung geprüft werden.“

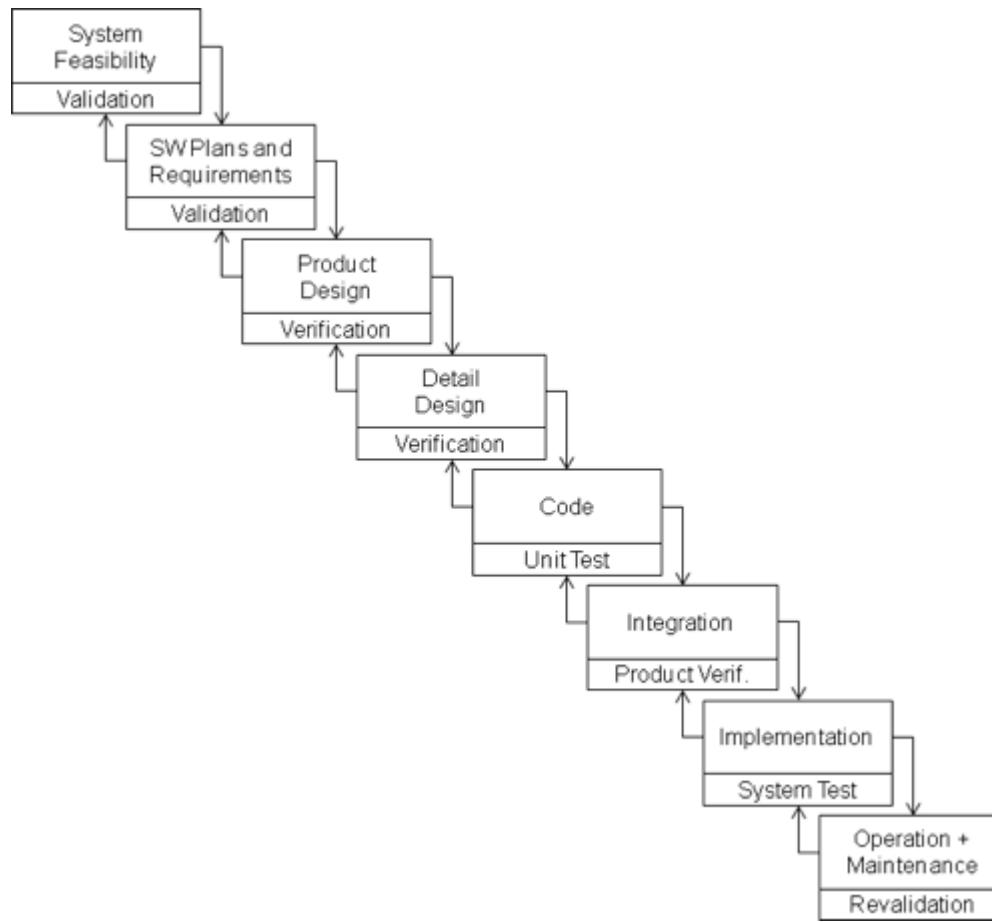
Quelle: Balzert, H. Lehrbuch der Software-Technik – Software-Entwicklung, 2. Auflage, 2001, S. 54

Es gibt verschiedene Arten von Vorgehensmodellen, die unterschiedliche Aspekte wie Vollständigkeit, Schnelligkeit der Auslieferung, Qualitätssicherung verstärkt in den Mittelpunkt stellen. Bekannte Vorgehensmodelle sind:

- Wasserfall-Modell
- Spiral-Modell
- Inkrementelles-iteratives Modell
- Agiles Modell

# Wann? - Vorgehensmodell (5)

## Wasserfall-Modell (nach Boehm)

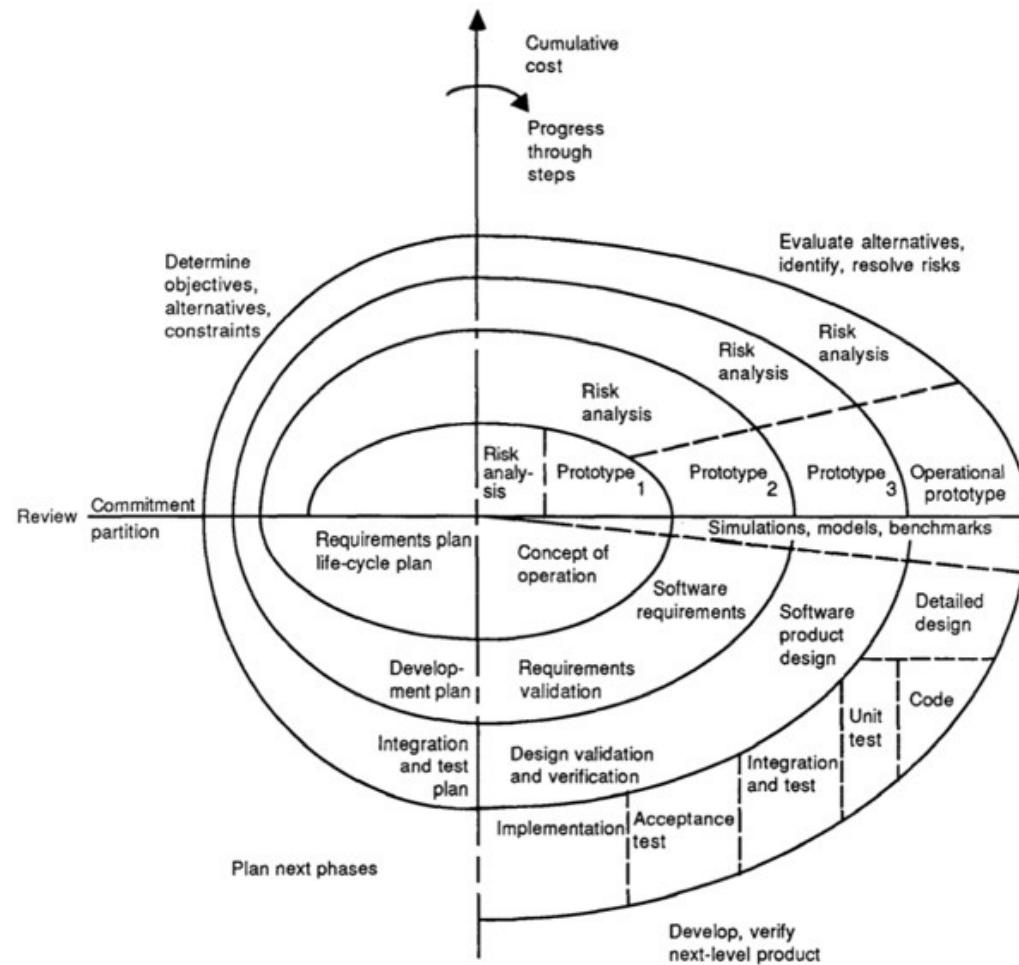


Quelle: Enzyklopädie der Wirtschaftsinformatik – Spiralmodell;

<http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/>

Vorgehensmodell/Wasserfallmodell/; Zugriff: 18.11.2019

## Spiral-Modell



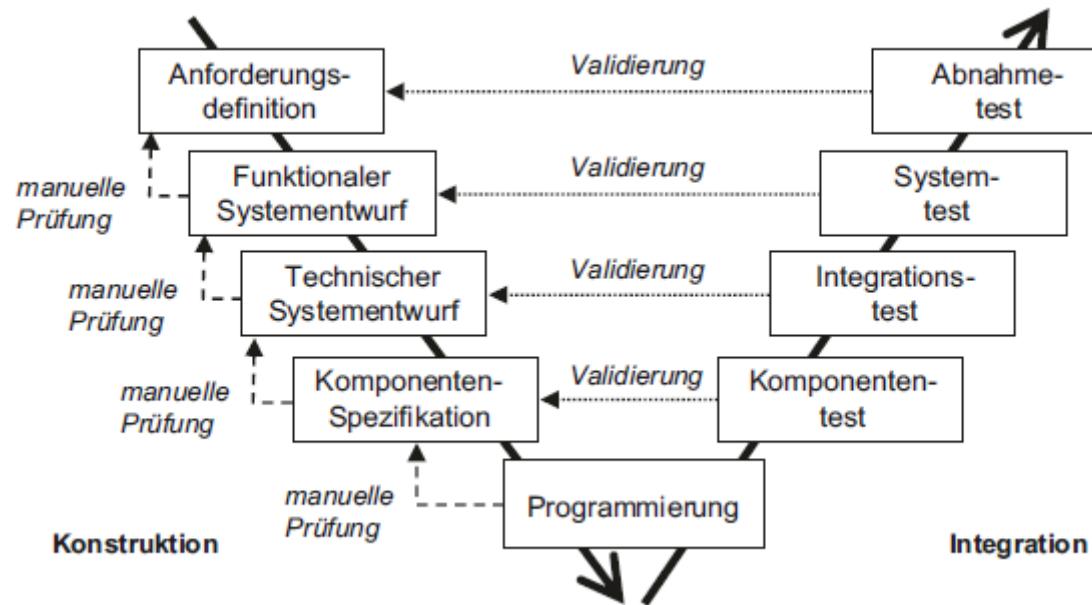
Quelle: Enzyklopädie der Wirtschaftsinformatik – Spiralmodell;

<http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/>

Vorgehensmodell/Spiralmodell/; Zugriff: 18.11.2019

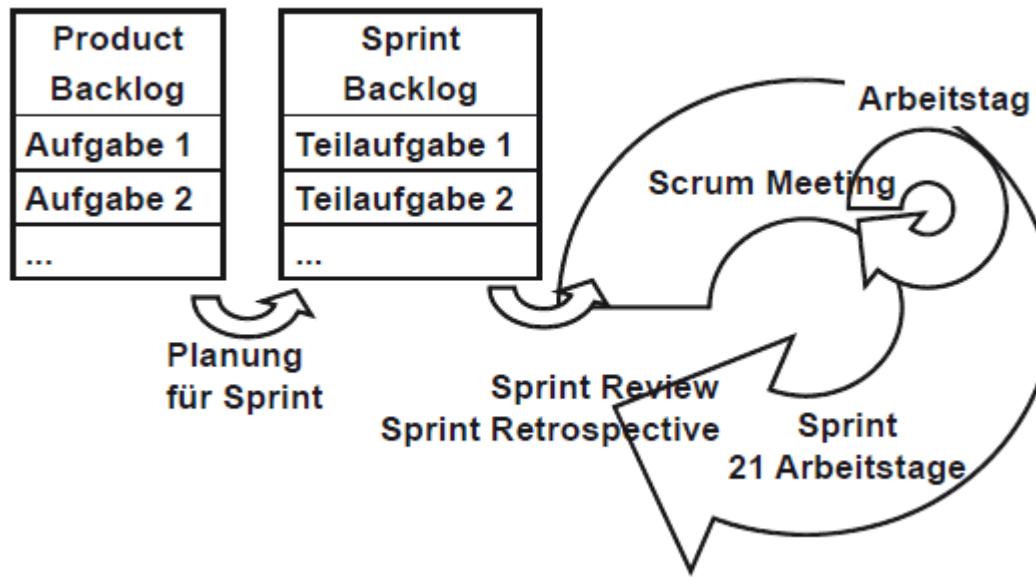
# Wann? - Vorgehensmodell (7)

## V-Modell



Quelle: Kleuker, S., Grundkurs Software-Engineering mit UML, 2018

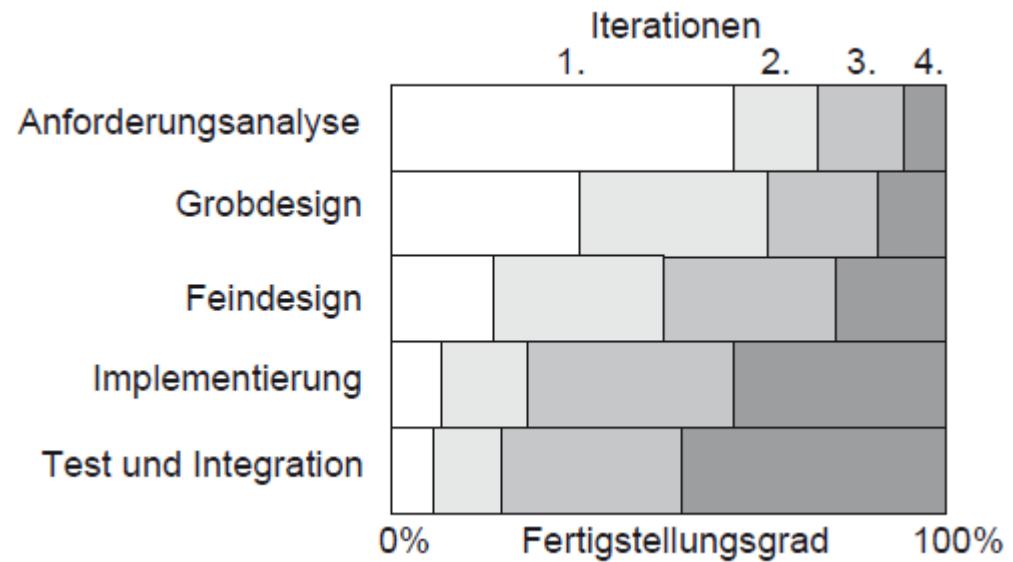
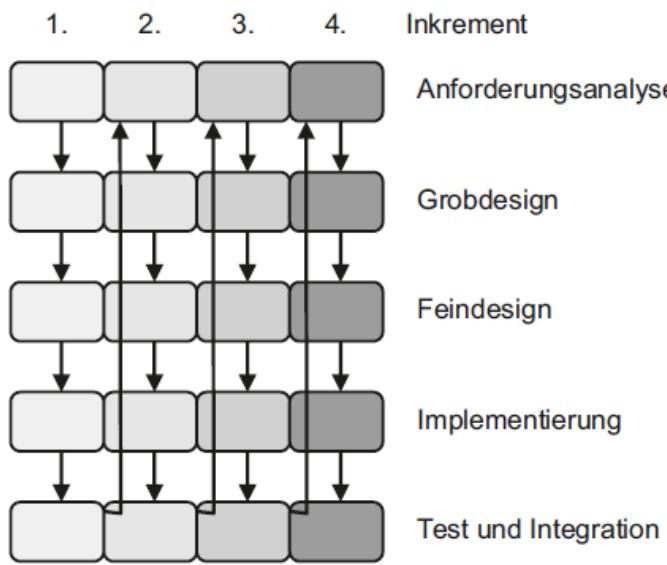
## Agiles-Modell (Scrum)



## Rollen

- Product owner: definiert, welche Aufgaben die zu erstellende Software übernehmen soll  
Scrum master: sorgt für die Einhaltung der Scrum Regeln  
Entwicklungsteam: erstellt die Software

## Inkrementelles-Iteratives Vorgehensmodell



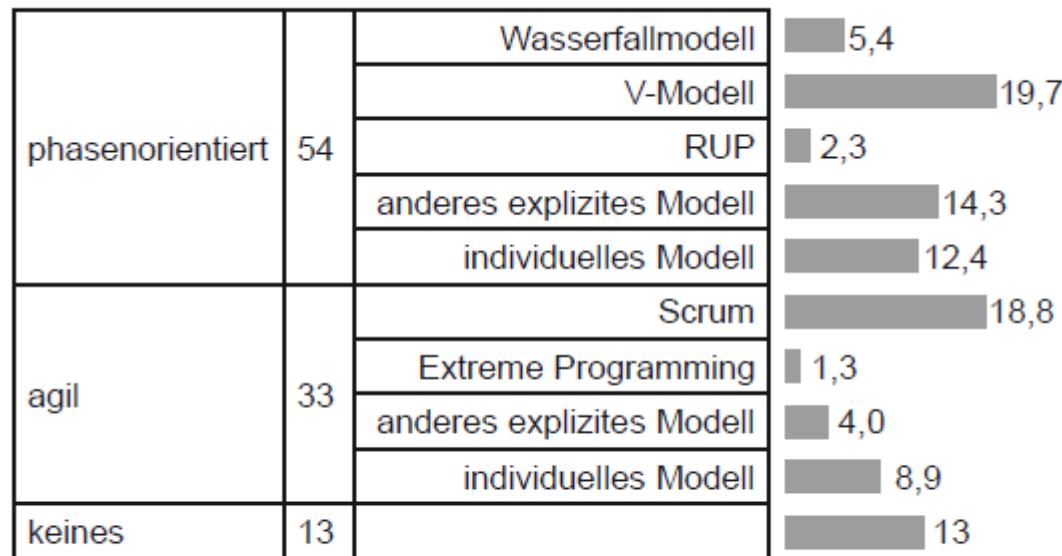
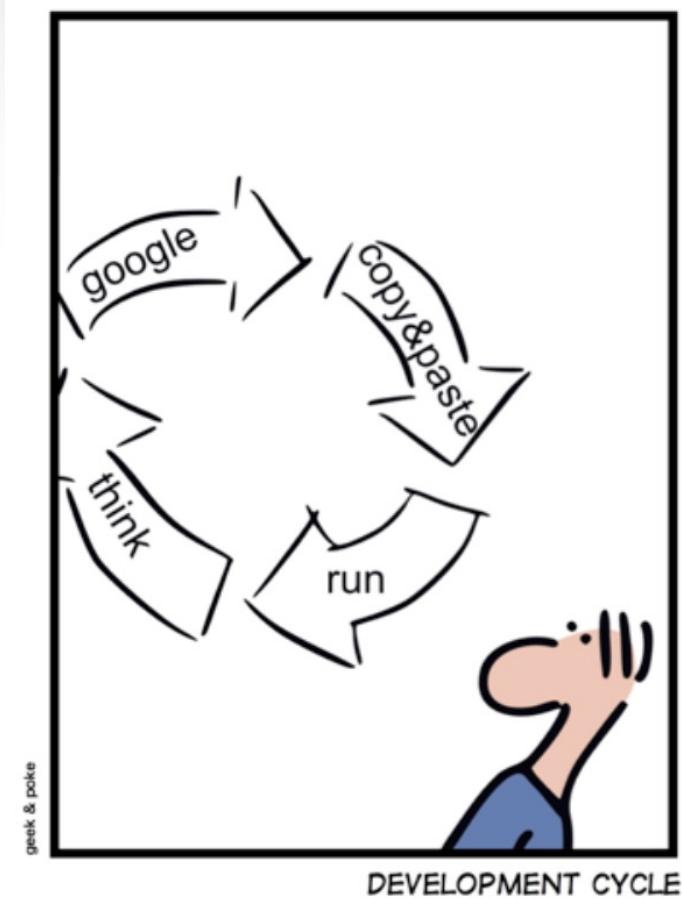


Abb. 12: Eingesetzte Vorgehensmodelle 2011 (in Prozent)

Quelle: Kleuker, S., Grundkurs Software-Engineering mit UML, 2018



Quelle: geek & poke; <http://geekandpoke.typepad.com/geekandpoke/2012/05/development-cycle.html> (Zugriff: 15.11.2016)

# Was?



Rene Magritte, Dies ist keine Pfeife, 1929

Quelle: [http://abcphil.phil-splitter.com/html/dies\\_ist\\_keine\\_pfeife.html](http://abcphil.phil-splitter.com/html/dies_ist_keine_pfeife.html); zugriff: 02.11.2015

„Alle Erkenntnis ist Erkenntnis in Modellen oder durch Modelle“

H. Stachowiak

Ein Modell ist ein **vereinfachtes Bild der Realität** (der „Welt“), in dem alle als **wesentlich** angesehen **Eigenschaften** des Systems erfasst wurden.

Nach der **allgemeinen Modelltheorie** (Stachowiak (1973)) können Modelle 6 Hauptgruppen zugeordnet werden:

- **Graphische Modelle**
  - **Bildmodelle** (vorwiegend anschaulich)
  - **Darstellungsmodelle** (brauchen Zeichenerklärung)
- **Technische Modelle**
  - **Physikotechnische Modelle** (vorwiegend hergestellt)
  - **Bio-psycho- und soziotechnische Modelle** (zumeist vorgefunden; werden aber manipuliert)
- **Semantische Modelle**
  - **Interne Modelle** (perzeptive und kognitive)
  - **Externe Modelle** (Zeichen-Modelle; Kommunikationssysteme)

Die Informatik arbeitet vorwiegend mit Darstellungs- und externen semantischen Modellen.  
04.07.24

Zusätzlich weisen Modelle nach Stachowiak (1973) 3 wesentliche Eigenschaften auf:

- **Abbildungsfunktion**

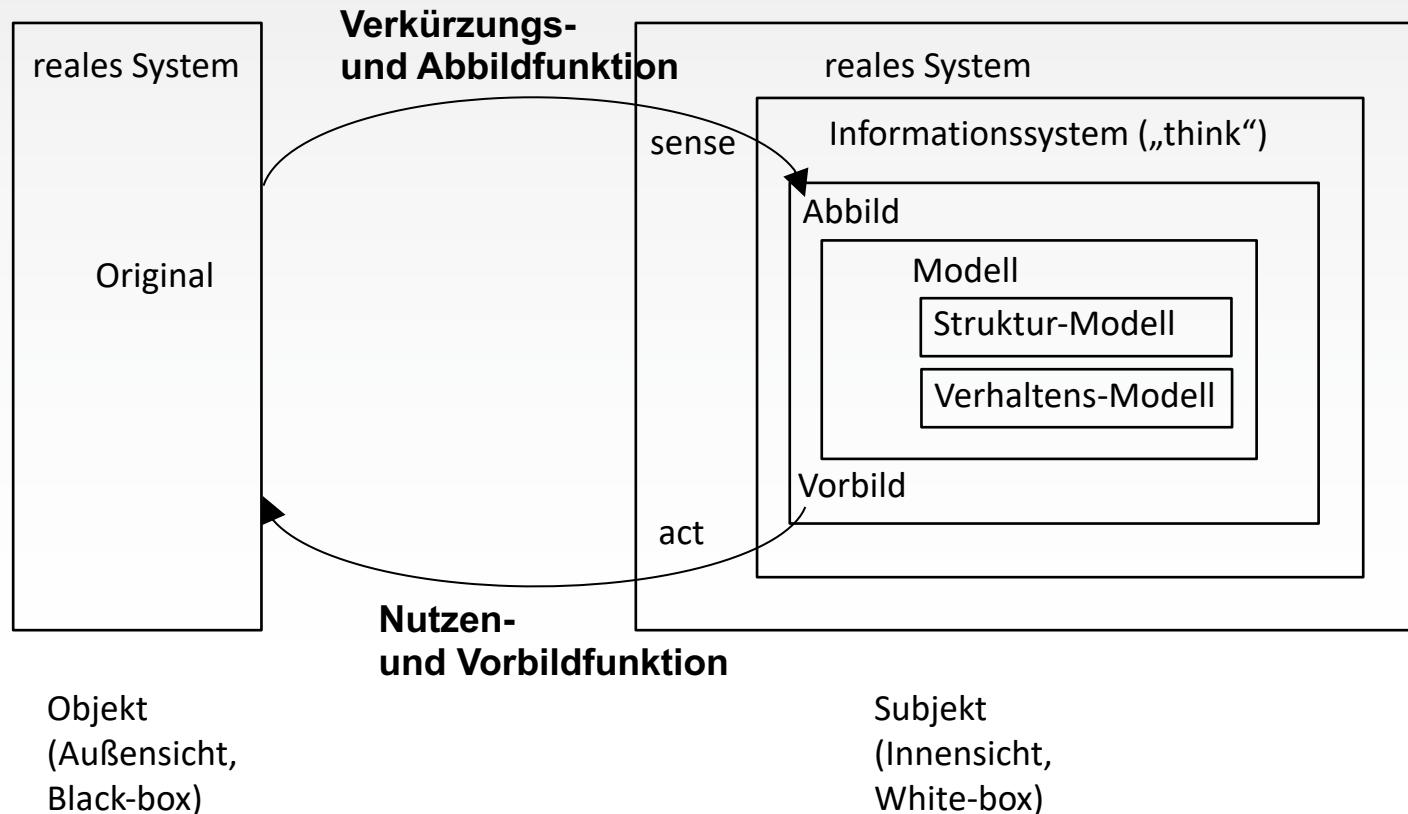
Es gibt eine (logische) Beziehung zwischen betrachteter „Welt“ und dem Modell. Ein Modell ist damit ein „**Abbild von etwas**“ oder ein „**Vorbild für etwas**“. D.h. entweder ist die Realität schon gegeben (**deskriptiv**) (Landschaft – Landkarte) oder aber sie wird hierdurch erst erschaffen (**präskriptiv**) (Konstruktionszeichnung – Produkt). Die betrachtete „Welt“ kann selbst auch ein Modell sein. Es kann verschiedene Modelle zu einer und derselben betrachteten „Welt“ geben.

- **Verkürzungsfunktion**

**Nicht alle Eigenschaften** der „Welt“ werden berücksichtigt, sondern nur für die Aufgabe relevante. Welche Eigenschaften relevant sind, ist **subjektiv**, d.h. vom Betrachter der „Welt“ abhängig. Da aufgrund der Komplexität der „Welt“ nicht alle Eigenschaften berücksichtigt werden können, wird das Prinzip der Abstraktion verwendet um Modelle für eine gegebene Aufgabe zu erstellen.

- **Pragmatische Funktion**

Modelle dienen einem **Zweck**, d.h. sie werden aufgrund ihres Nutzens (Dokumentation, Kommunikation) für unterschiedliche Rollen beurteilt. Modelle werden u.a. angefertigt, da Änderungen am Original z.B. nicht durchführbar oder zu teuer sind (z.B. Konstruktion eines Autos). Oder aber das Original zu komplex ist (z.B. Landschaft – Landkarte).



## The Uses of Models (REDCAPE)

**Reason:** To identify conditions and deduce logical implications.

**Explain:** To provide (testable) explanations for empirical phenomena.

**Design:** To choose features of institutions, policies, and rules.

**Communicate:** To relate knowledge and understandings.

**Act:** To guide policy choices and strategic actions.

**Predict:** To make numerical and categorical predictions of future and unknown phenomena.

**Explore:** To investigate possibilities and hypotheticals.

Quelle: Page, S.; The Model Thinker: What You Need to Know to Make Data Work for You, 2018

- **black-box view**

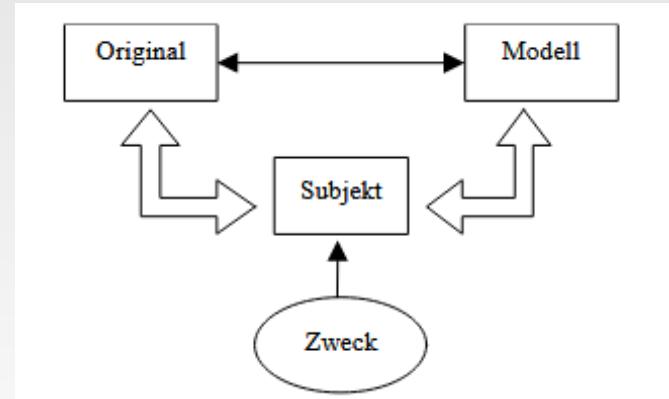
„A very common abstraction technique is to model the system as a black-box, which only exposes the features of the system that are visible from an external observer and hides the internal details of the design. This includes externally visible behavior and other physical characteristics... .“

- **white-box view**

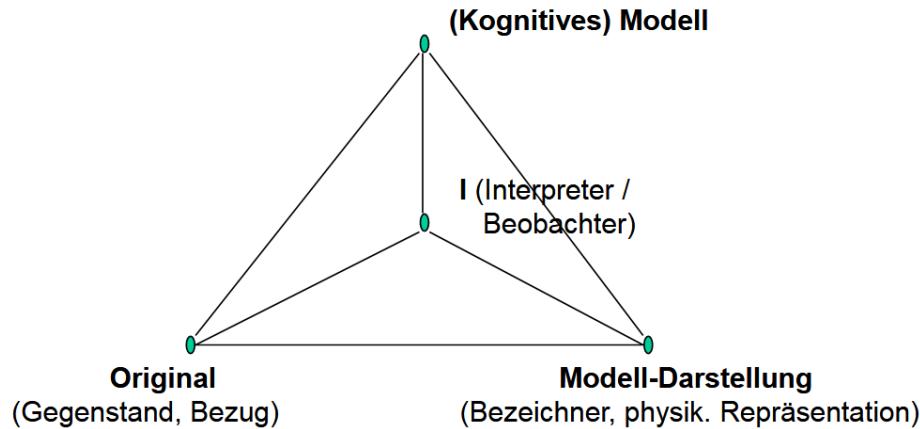
„A white-box model of a system ... shows the internal structure and displays the behavior of the system.“

Quelle: SEBok – Guide to the Systems Engineering Body of Knowledge  
[https://www.sebokwiki.org/wiki/System\\_Modeling\\_Concepts](https://www.sebokwiki.org/wiki/System_Modeling_Concepts); Zugriff: 11.11.2019

# Was? - Modell (7)

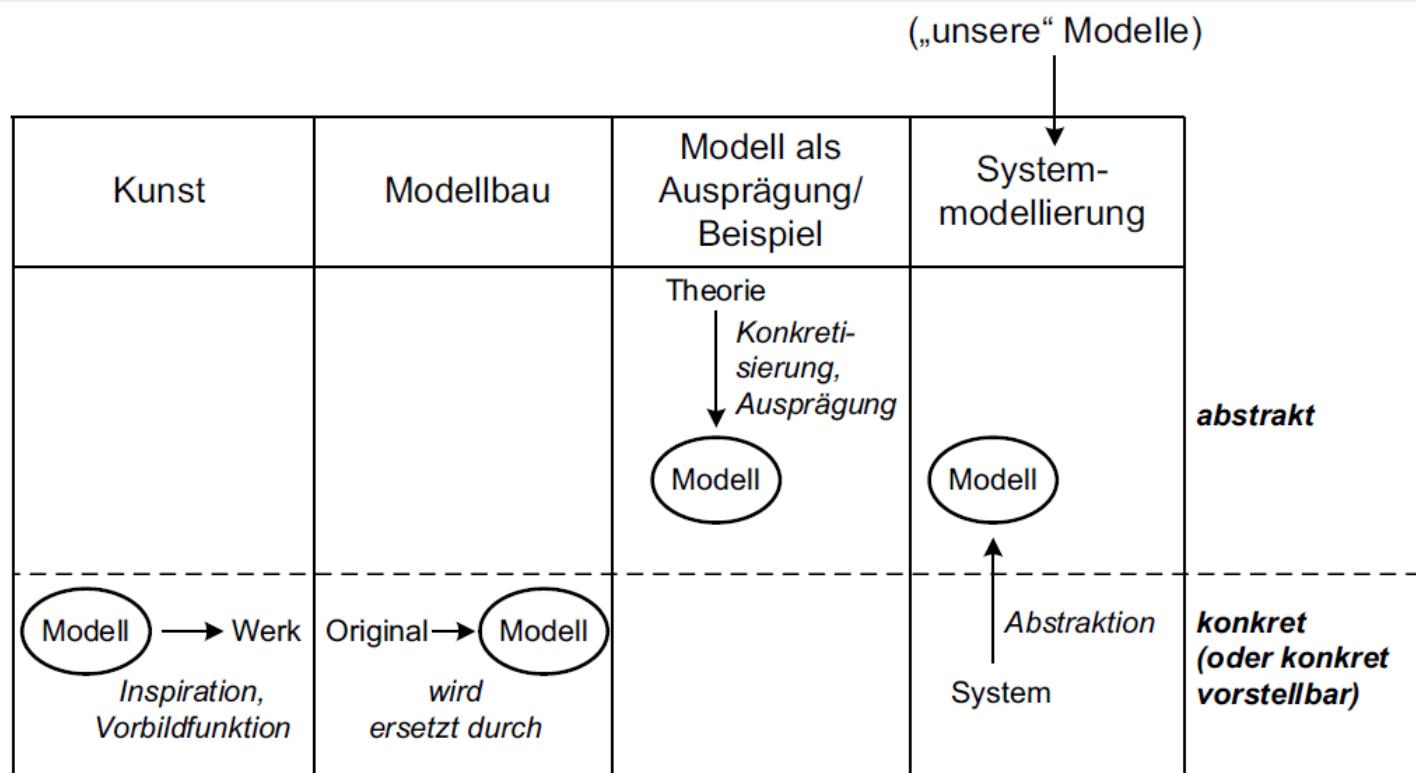


Quelle: Thomas, M.; Die Vielfalt der Modelle in der Informatik, 2008

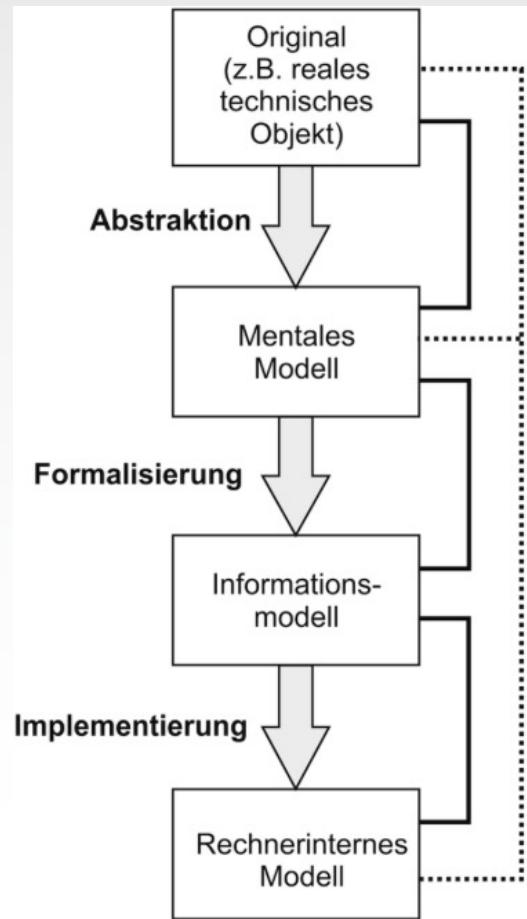


Quelle: Hesse, W.; Modellierung dynamischer und adaptiver Systeme, WiSo 2014/15  
[https://www.pst\\_ifi.lmu.de/Lehre/wise-15-16/modas/modas2.pdf](https://www.pst_ifi.lmu.de/Lehre/wise-15-16/modas/modas2.pdf); Zugriff: 29.10.2018

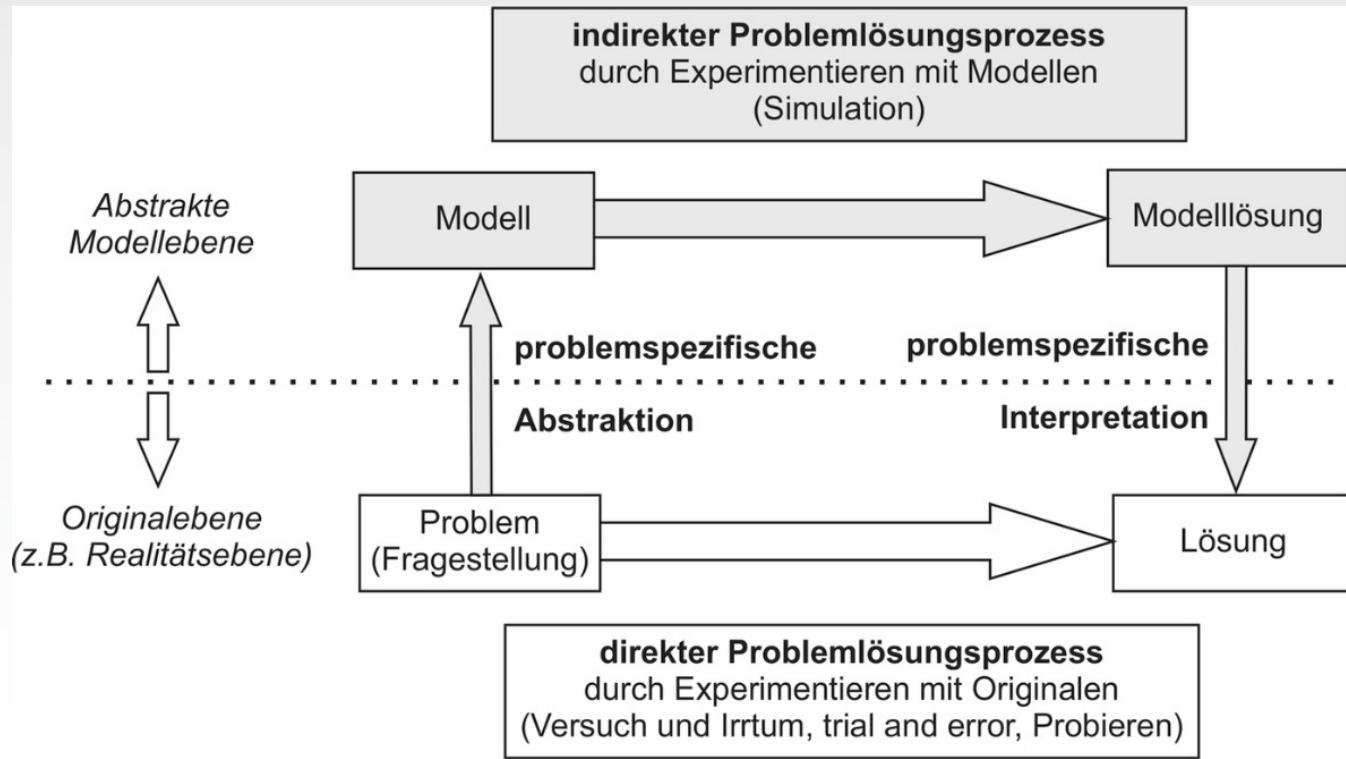
# Was? - Modell (8)



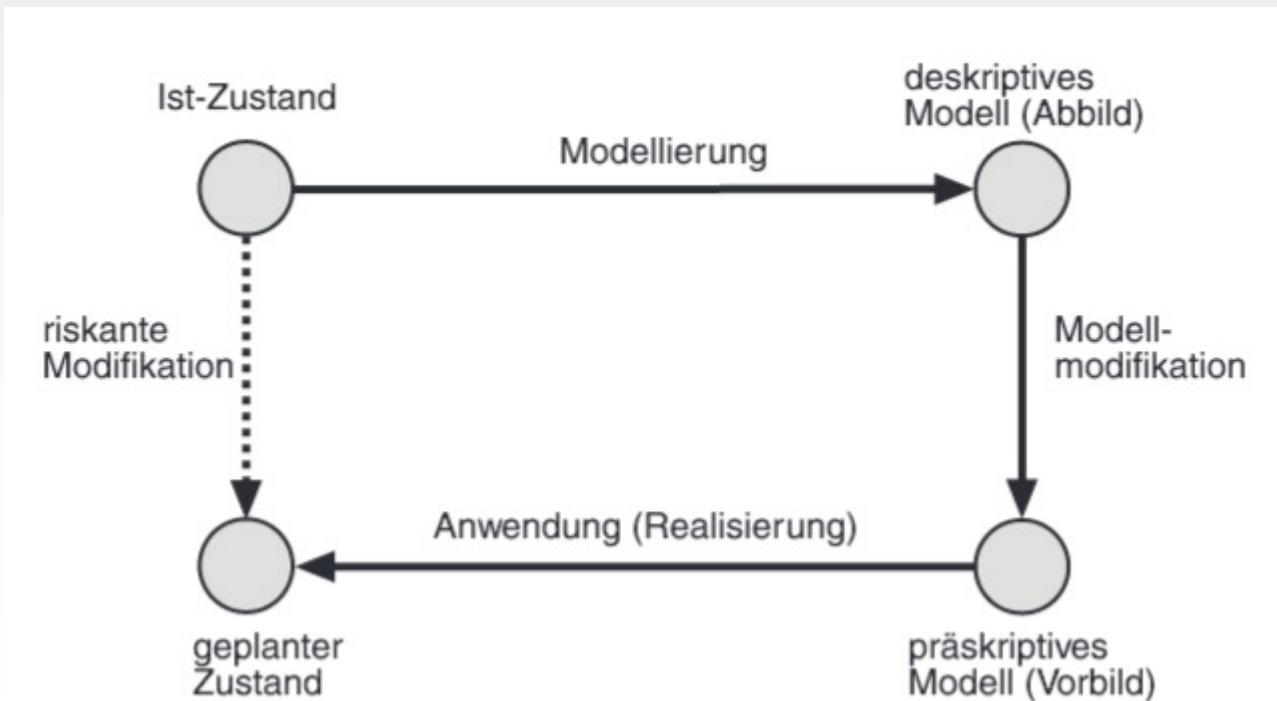
Quelle: Tabeling, P.; Softwaresysteme und ihre Modellierung, Springer 2006



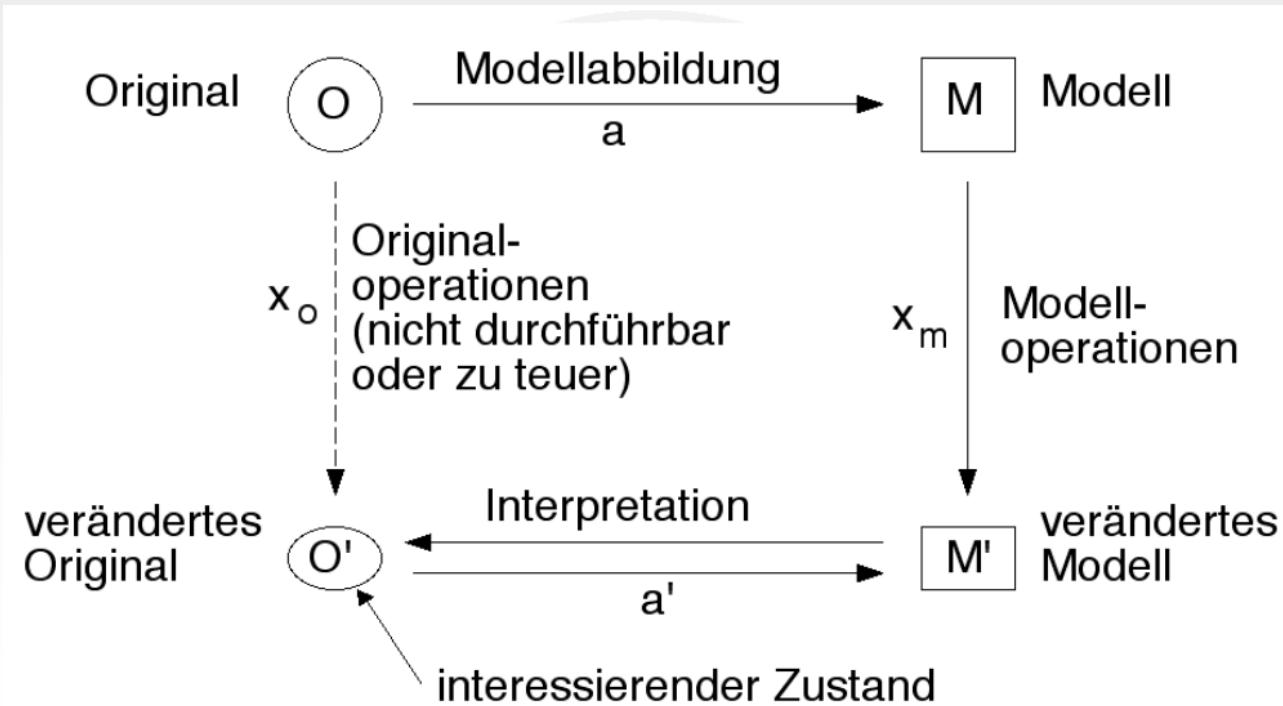
Quelle: Vajna, S.; et al.; CAx für Ingenieure – Eine praxisbezogene Einführung, 3. Aufl., 2018



Quelle: Vajna, S.; et al.; CAx für Ingenieure – Eine praxisbezogene Einführung, 3. Aufl., 2018

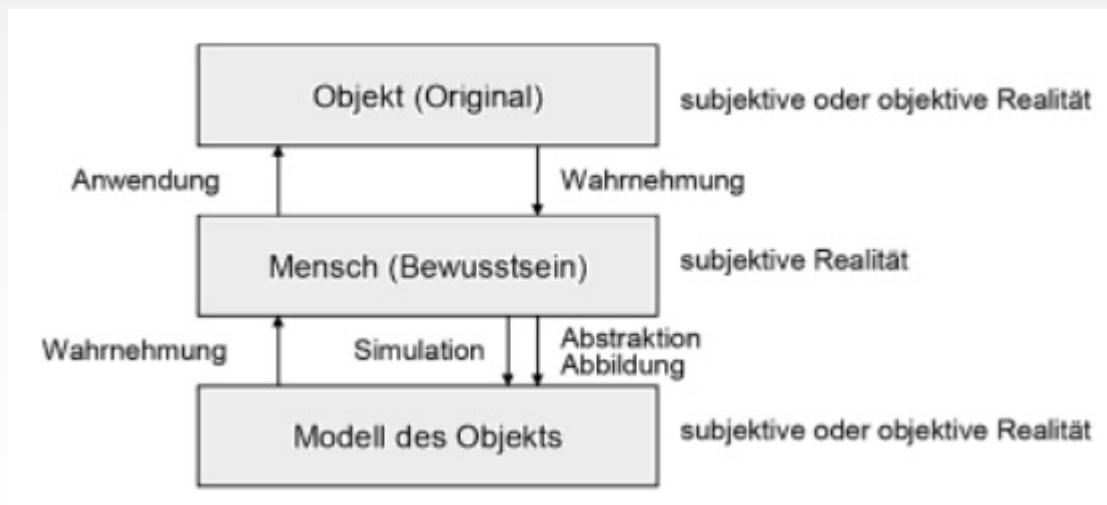


Quelle: Ludewig, J.; Licher, H, Software Engineering: Grundlagen, Menschen, Prozesse, Techniken; 2. Aufl, dpunkt, 2010

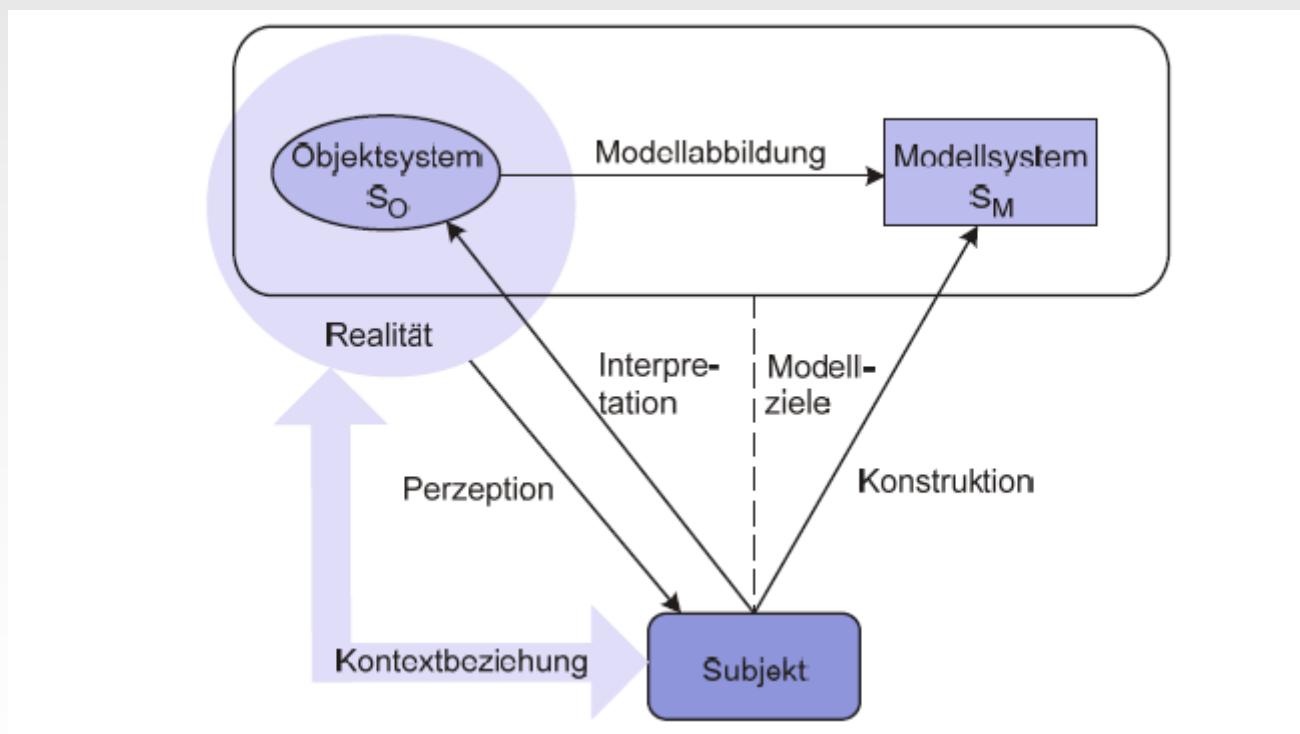


Quelle: Glinz, M, Einführung in die Modelltheorie, 2005

[https://files\\_ifi\\_uzh\\_ch/rerg/amadeus/teaching/courses/inf\\_IIfs10/inf\\_IIf\\_kapitel\\_02.pdf](https://files_ifi_uzh_ch/rerg/amadeus/teaching/courses/inf_IIfs10/inf_IIf_kapitel_02.pdf); Zugriff: 24.11.2014



Quelle: Krallmann, H. et al; Systemanalyse im Unternehmen: prozessorientierte Methoden der Wirtschaftsinformatik, 6.Aufl, 2013



„Das konstruktivistische Modellierungsverständnis macht deutlich, dass Modellbildungen notwendigerweise subjektiven Einflüssen unterliegen. Es ist daher ein wichtiges Anliegen der Modellierung, subjektive Einflüsse zu begrenzen und sichtbar zu machen und somit Modelle als Grundlage für die Kommunikation zwischen unterschiedlichen Subjekten nutzen zu können.“

Quelle: Ferstl, O., Sinz, E.: Grundlagen der Wirtschaftsinformatik, 7.Aufl., Oldenbourg, 2015

Systemanalyse kann als „Arbeiten an Modellen“ angesehen werden. Hierbei kommen in Abhängigkeit der entsprechenden Aufgaben/Rollen unterschiedliche Modelle zum Einsatz.

Für die Modellierung im Softwareentwicklungsbereich wird die „**Unified Modeling Language (UML)**“ benutzt.

Jede an einem Softwareprojekt beteiligte Rolle benötigt ihr spezielles Modell. So gibt es z.B. für AnwenderInnen das **Fachmodell**, für ArchitektInnen ein **Architekturmodell**, für ProgrammiererInnen ein **Programmiermodell**, für DatenbankadministratorInnen ein **Datenmodell/Datenbankmodell**.

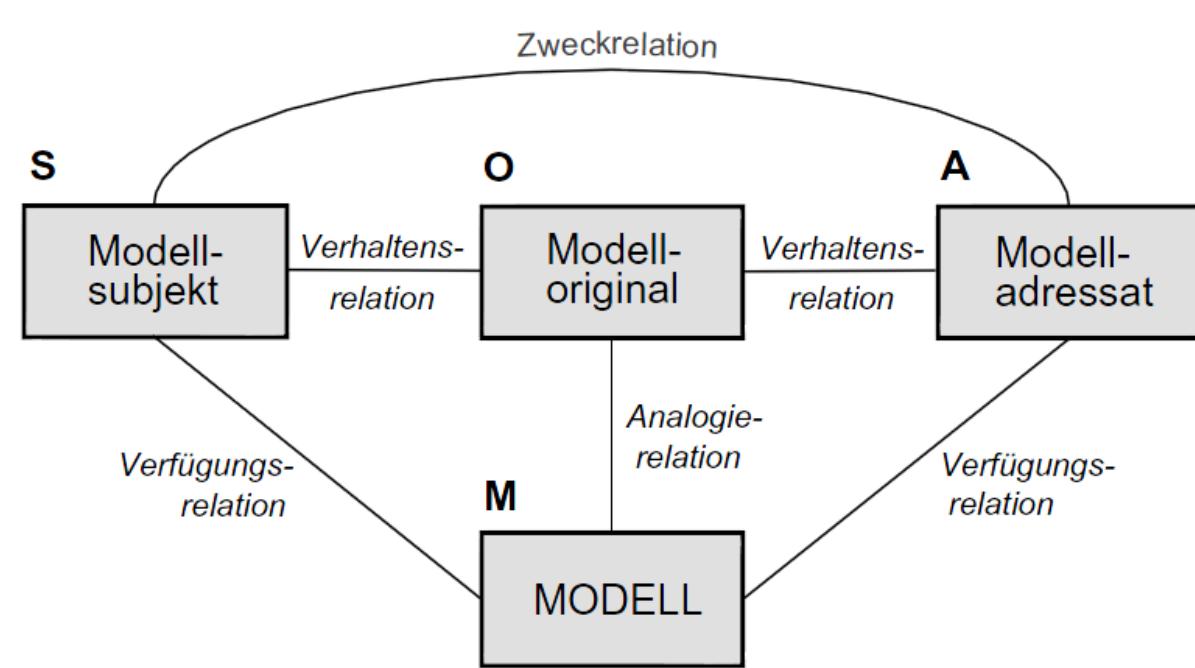
Ziel der Systemanalyse ist es in einem **Anwendungsbereich (Domäne)** wie z.B. Wirtschaft, Umwelt, Medizin die wesentlichen Begrifflichkeiten und ihre Beziehungen herauszuarbeiten (strukturelles Modell) bzw. die wesentlichen funktionalen Abläufe zu finden (Verhaltensmodell). Zu diesem Zweck werden sowohl **Strukturmodelle** als auch **Verhaltensmodelle vom Systemanalytiker für den Anwendungsbereich** erstellt. Diese Fachmodelle bilden die Grundlage für alle an einem Softwareprojekt beteiligte Rollen. Alle weiteren Modelle sind Spezialisierung dieser Fachmodelle.

Merkmal	Merkmalsausprägung			
Beschreibungs-sicht	Daten	Funktionen	Organisation	
	Objekte			Prozesse
Beschreibungs-ebene	Fachkonzept		DV-Konzept	Implementierungs-konzept
Geltungs-anspruch	Istmodell		Sollmodell	Idealmodell
Inhaltliche Individualität	Unternehmens-modell		Referenzmodell	Metamodell
Sprache	Natürliche Sprache		Diagramm-sprache	Skriptsprache
Abstraktions-grad	Ausprägungs-ebene	Typebene	Metaebene	Metameta-ebene

Quelle: Krallmann, H. et al; Systemanalyse im Unternehmen: prozessorientierte Methoden der Wirtschaftsinformatik, 6.Aufl, 2013

„Modell ist stets Modell-wovon-wozu-für-wen.“

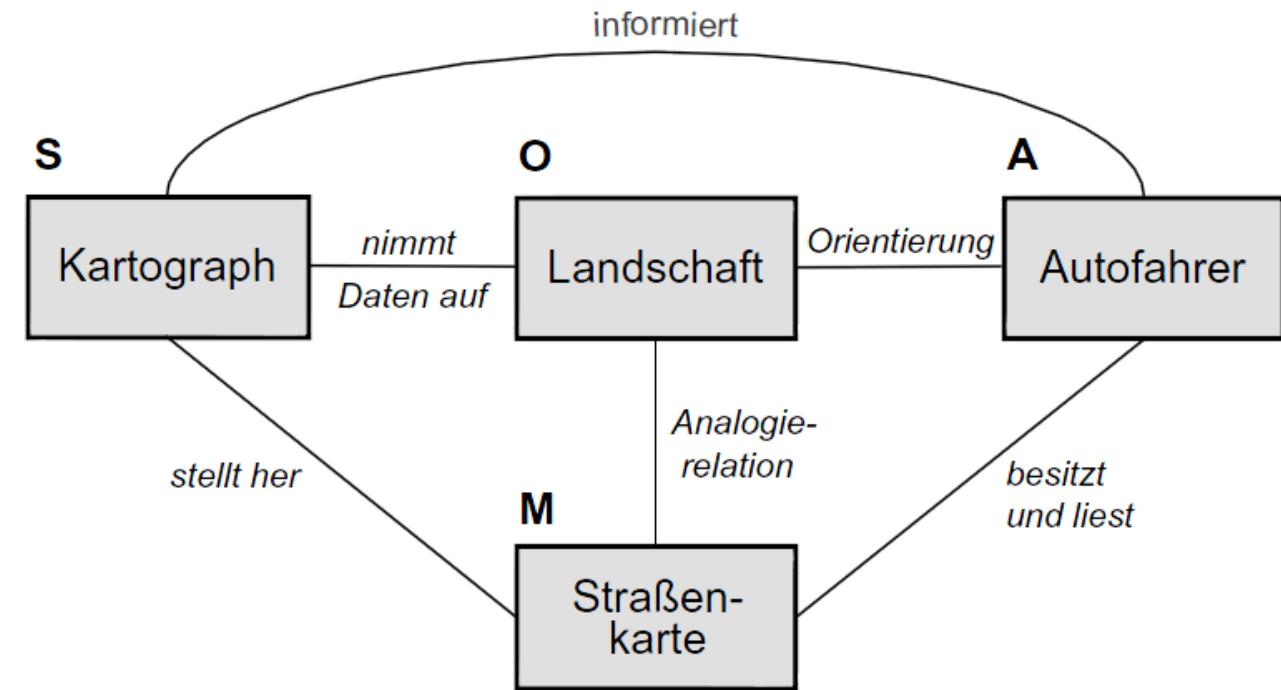
W. Steinmüller



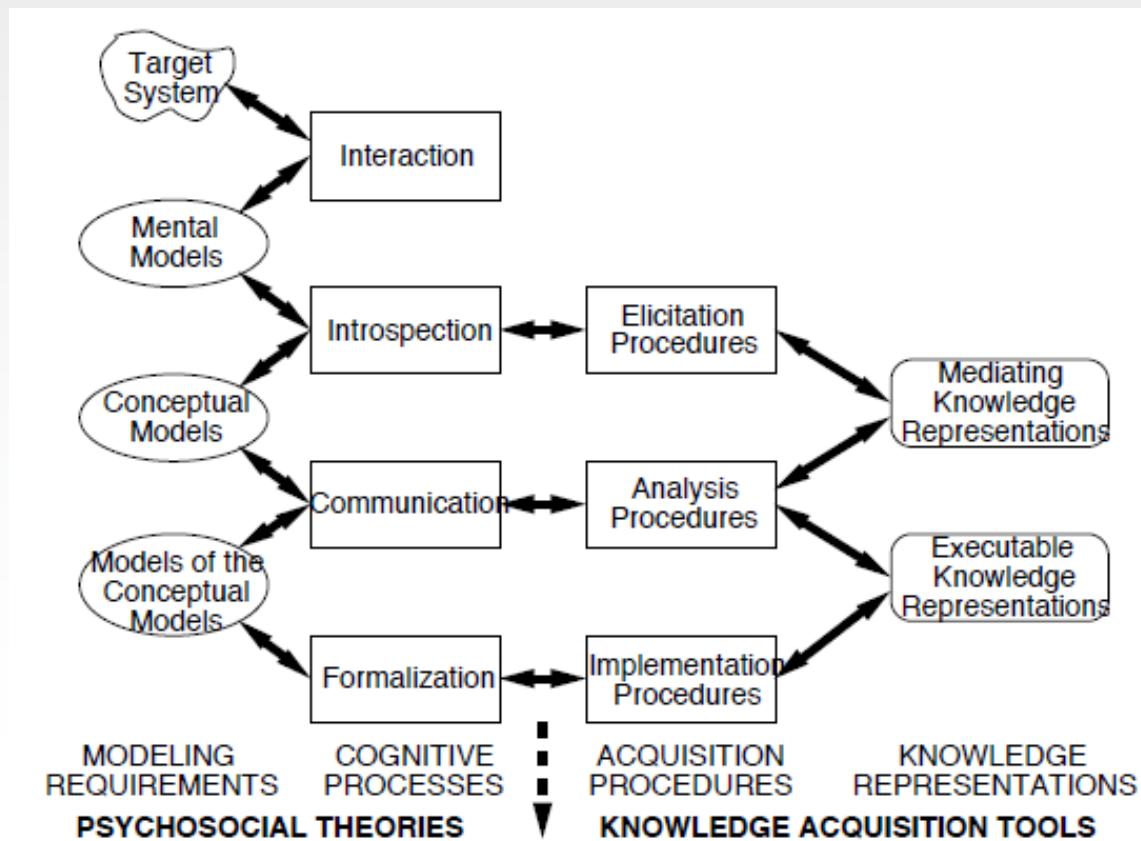
**SOMA**-Diagramm: **S**ubjekt – **O**riginal – **M**odell – **A**dressat

# Was? - Modell (18)

Beispiel: Straßenkarte als Modell



## Modelle und ihre Beziehung untereinander

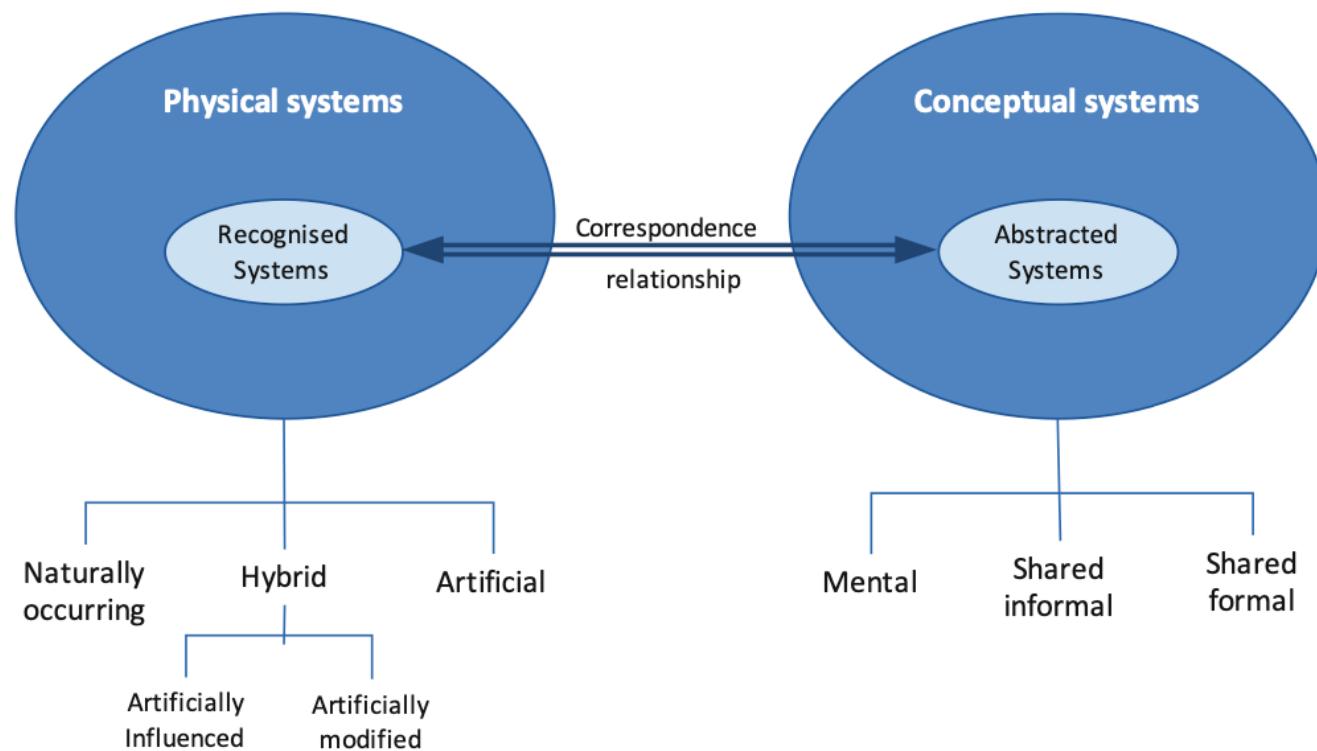


Quelle: Bredshaw, J.; Boose, J.; Mediating Representations for Knowledge Acquisition;  
<http://www.jeffreymbradshaw.com/publications/Mediating%20Reps%20for%20KA.pdf>; Zugriff: 03.11.2014

„We use the term system as a collection of interconnected parts. Modeling is a way to deal with complexity by ignoring irrelevant details. We use the term model to refer to any abstraction of the system.”

Quelle: Bruegge, B., Dutoit, A., Object-oriented Software Engineering, 3rd ed., 2010

## Taxonomy of physical and conceptual systems/models



## Modelle und ihre Beziehung untereinander

- mentales Modell

Aufgrund der Interaktion mit der physischen und geistigen Umwelt entsteht im Kopf eines Menschen ein sogenanntes „mentales“, also geistiges Modell der Umwelt durch Anwendung von Abstraktion.

- konzeptionelles Modell – fachliche Modelle

In einem konzeptionellen Modell werden die Elemente (und deren Beziehungen), die ein Beobachter der Umwelt wahrnimmt, in Form von Konzepten ausgedrückt. Das konzeptionelle Modell ist ein mentales Modell, welches externalisiert wurde, d.h. es wurde auf einem physischen Medien (z.B. Papier) dargestellt. Konzeptionelle Modelle in der Informatik werden u.a. dazu verwendet um die Fachbegriffe und Beziehungen aus einem fachlichen Anwendungsbereich (z.B. Wirtschaft) darzustellen. Ein Fachmodell ist also eine Form eines konzeptionellen Models.

- „abgeleitete Modelle“ – technische Modelle

Aus dem konzeptionellen Modell bzw. dem Fachmodell lassen sich für die Informatik u.a. technikbezogene Modelle wie Oberflächen- und Datenbankmodell ableiten. Diese Modelle „repräsentieren“ den Inhalt des Fachmodells unter Zuhilfenahme von Technik.

„It is widely accepted in the cognitive science and psychology literature that people develop and use internal representations, i.e., ‘mental models’, of external reality that allow them to interact with the world. ... Mental models are conceived of as a cognitive structure that forms the basis of reasoning, decision making ... “

“A mental model is a simplified representation of reality that allows people to interact with the world.”

Jones, N.; Mental Models: An Interdisciplinary Synthesis of Theory and Methods, 2011

<https://www.jstor.org/stable/pdf/26268859.pdf?refreqid=excelsior%3Ad0b29d92b62e6cd99d4874976d2d0c57>; Zugriff 29.10.2019

„A conceptual model – sometimes called business or domain model – describes the users' world in a modeling language independently of implementation technology and constraints. Conceptual modeling is applied in the early phases of information system analysis and design.“

Birkmeier, D. et. al.; A method to support a reflective derivation of business components from conceptual models, 2012

„A key activity in systems analysis is to conceptualize the domain under study and represent it in one or more conceptual models. Conceptual models serve as representations of user's perceptions of a static and dynamic phenomena in a domain and are usually graphical in nature.“

Burton-Jones, A., Meso, P.; Conceptualizing Systems for Understanding: An Empirical Test of Decomposition Principles in Object-Oriented Analysis, 2006

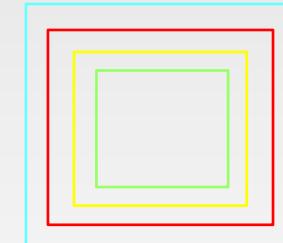
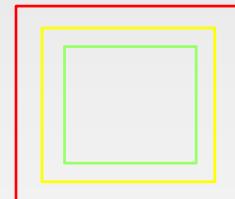
„A conceptual model should reflect knowledge about the application domain rather than about the implementation of the information system.“

Wand, Y.; Monarchi, D., Parsons, J.; Woo, C.; Theoretical foundations for conceptual modelling in information systems development, Decision Support Systems 15, 1995

“A conceptual model is a representation of an application domain intended to support the analysis, documentation and communication of knowledge about the domain. ”

# Was? - Modell (25)

## Modelle und ihre Beziehung untereinander (Softwareentwicklungslebenszyklus)



Fachmodell

Entwurfsmodell

Realisierungsmodell

Betriebsmodell

Fachbegriffe und  
Beziehungen  
(Fachmodule)

Fachbegriffe und  
Beziehungen  
(Fachmodule)

Fachbegriffe und  
Beziehungen  
(Fachmodule)

Fachbegriffe und  
Beziehungen  
(Fachmodule)

Entwurfsmuster +  
UI-Modell /  
DB-Modell

Entwurfsmuster +  
UI-Modell /  
DB-Modell

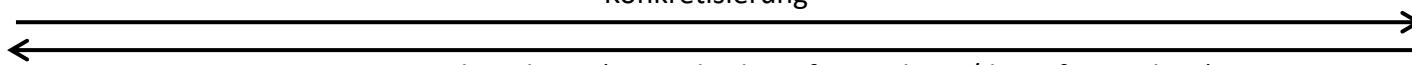
Entwurfsmuster +  
UI-Modell /  
DB-Modell

Datenstrukturen &  
Algorithmen,  
Frameworks

Datenstrukturen &  
Algorithmen,  
Frameworks

Ausführungs-  
umgebung

Konkretisierung

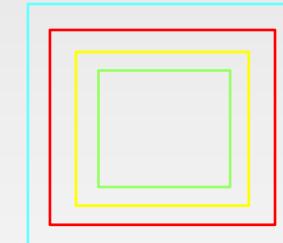
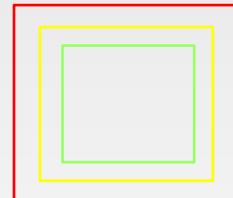
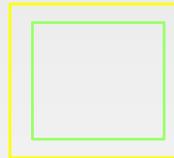


Abstraktion (aus Sicht der Informatikerin/des Informatikers)

© Thomas Slotos

# Was? - Modell (26)

## Modelle und ihre Beziehung untereinander (Softwareentwicklungslebenszyklus)



Fachmodell

Entwurfsmodell

Realisierungsmodell

Betriebsmodell

z.B. Begriffe und Beziehungen innerhalb und zwischen den Domänen „Küche“ und „Speisesaal“ eines „Restaurants“

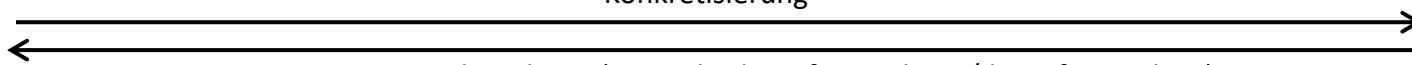
z.B. Begriffe und Beziehungen innerhalb und zwischen den Domänen „Küche“ und „Speisesaal“ eines „Restaurants“  
z.B. Wireframes für GUI, ER-Modell für rel. DBMS

z.B. Begriffe und Beziehungen innerhalb und zwischen den Domänen „Küche“ und „Speisesaal“ eines „Restaurants“  
z.B. Wireframes für GUI, ER-Modell für rel. DBMS  
z.B. Search-Algorithmus auf B-Bäumen, Spring-Framework für Connectivity

z.B. Begriffe und Beziehungen innerhalb und zwischen den Domänen „Küche“ und „Speisesaal“ eines „Restaurants“  
z.B. Wireframes für GUI, ER-Modell für rel. DBMS  
z.B. Search-Algorithmus auf B-Bäumen, Spring-Framework für Connectivity

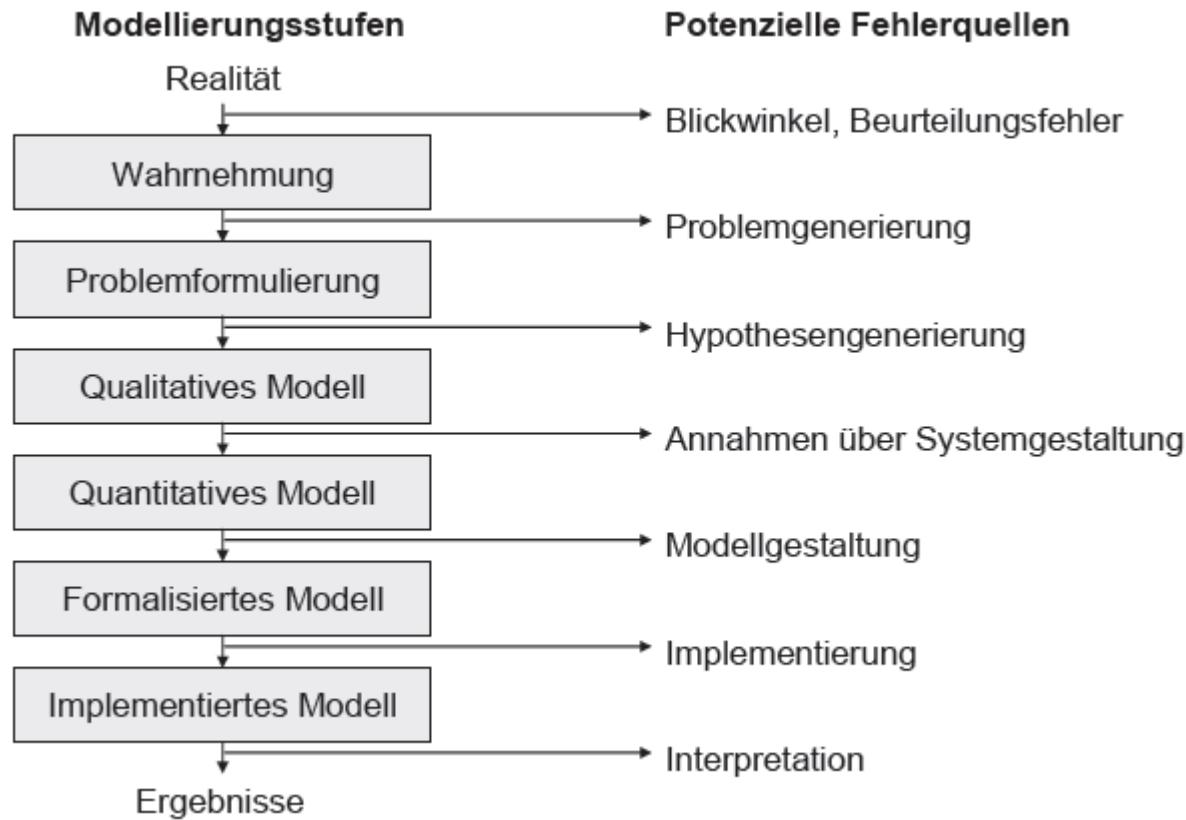
z.B. Docker / Kubernetes für Microservices, Anwendungen in der Cloud

Konkretisierung



Abstraktion (aus Sicht der Informatikerin/des Informatikers)

© Thomas Slotos



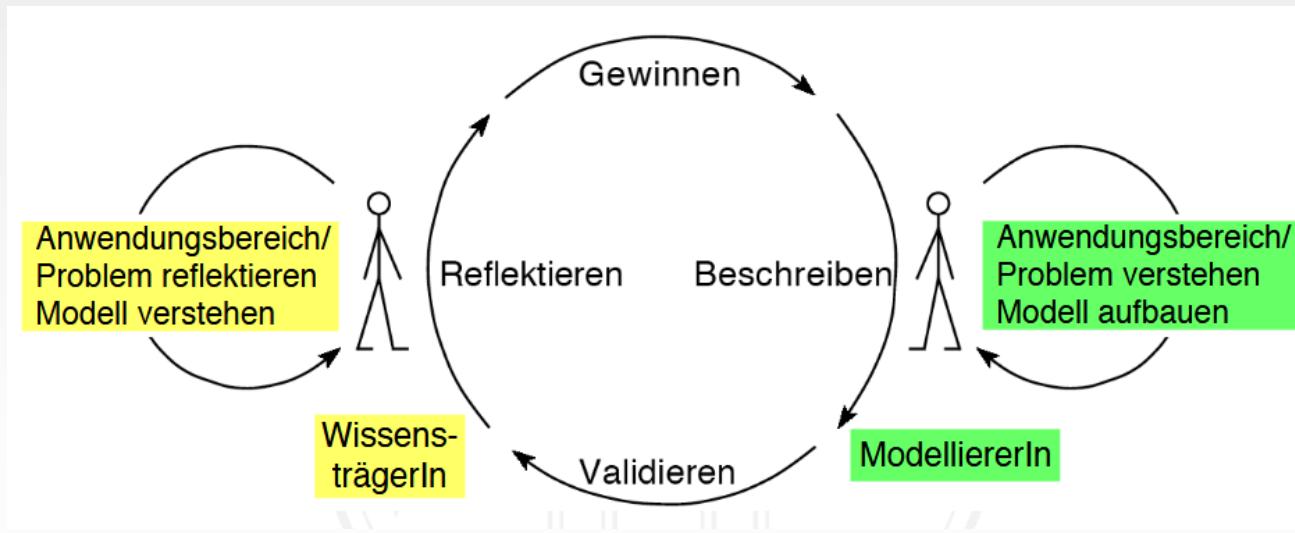
# Was? - Modell (28)

W-Fragen	Antworten	Beispiel (Kartographie)
Wer?	Subjekt / Stakeholder	Kartograph, Kartennutzer
Wozu?	Kosten-/Zeiteinsparung, Orientierung	Zeitersparnis wg. kürzester Weg von A nach B
Warum?	Original nicht verfügbar, zu groß, zu komplex	Geografisches Gebiet (z.B. Stadt) zu unüberschaubar
Was?	Verschiedene Modelltypen	Straßenkarte, Radfahrkarte
Womit?	Modellierungswerkzeug	Kartografieprogramm
Wieviele?	Views	Radfahrkarte 1:75.000 / Radfahrkarte 1:200.000
Wovon?	Original	Geografisches Gebiet (z.B. Stadt)
Wann?	Versionen	Radfahrkarte 1990, 2000, 2010, 2020
Wie?	Abstraktion, Abbildung	Radwege sind im wesentlichen auf Radfahrkarte (Gebäude kaum vorhanden)

„Just like the purpose of a good map is useful because it omits unnecessary details while representing only the essential features for the purpose of navigation, so is modeling an art of making appropriate assumptions for specific purposes, i.e. of finding the relevant level and units of abstraction essential for understanding the system ... Unlike the analogy to cartography, in science it is often the iterative process of modeling that matters, and not the model itself.“

Green, S.; Wolkenhauer, O.; Tracing Organizing Principles: Learning from the History of Systems Biology, 2013

Modellbildung als iterativer (wiederholender) Prozess

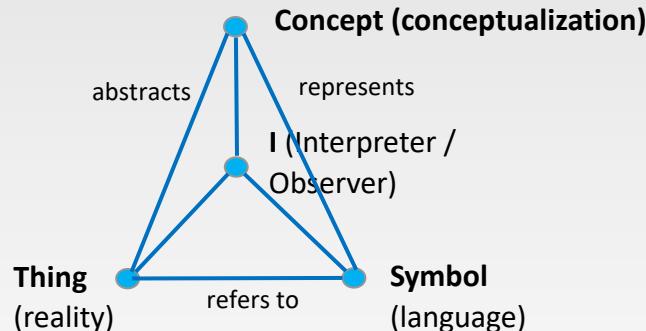


Modelle entstehen durch Verwendung einer **Sprache** (z.B. grafische, textuelle). Mit einer Sprache ist eine strukturierte Menge von Zeichen und damit bezeichneten begrifflichen Vorstellungen gemeint. **Zeichen** können Laute, Schrift, Symbole usw. sein. Ein System von Schrift- und Symbolzeichen zur Darstellung eines Modells wird **Notation** genannt.

Zwei Kommunikationspartner (Menschen oder Maschinen) **kommunizieren** durch den Austausch einer Menge an Zeichen. Für eine erfolgreiche Kommunikation ist ein gemeinsamer Zeichenvorrat (**Lexikalik**) sowie Regeln zur Bildung von Zeichenstrukturen (**Syntax**) notwendig. Die Bedeutung der Zeichen (**Semantik**) erfolgt durch die Zuordnung zu Begriffen. Man kann auch sagen, dass Kommunikationspartner Modelle basierend auf einer Notation austauschen. Die Notation im Bereich der angewandten Informatik stellt die Unified Modelling Language (UML) dar. Das Modell wird in einem Anwendungskontext mit einer bestimmten Zielsetzung erstellt. Aber auch bestehendes Wissen des Modellierers fließt in die Modellbildung ein. Diese Faktoren bilden die **Pragmatik** unter dem das Modell entsteht.

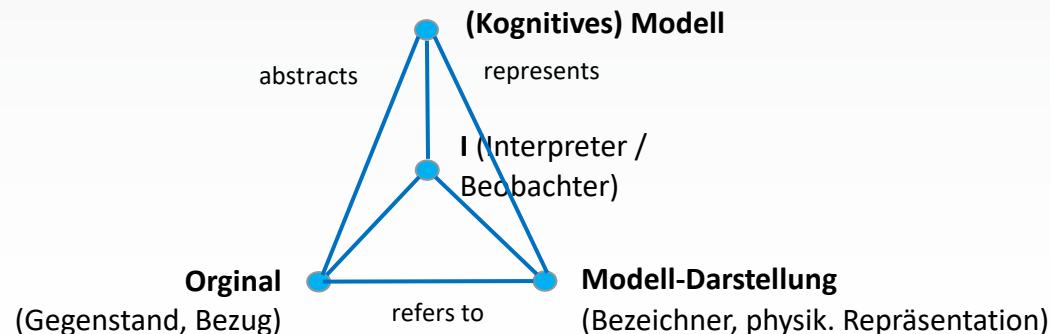
Basierend auf: Glinz, M, Einführung in die Modelltheorie, 2005  
[https://files\\_ifi\\_uzh\\_ch\\_rerg\\_amadeus\\_teaching\\_courses\\_inf\\_II\\_fs10\\_inf\\_II\\_kapitel\\_02.pdf](https://files_ifi_uzh_ch_rerg_amadeus_teaching_courses_inf_II_fs10_inf_II_kapitel_02.pdf); Zugriff: 24.11.2014

## Language



Quelle: (erweitert nach) Guizzardi, G.; On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models, 2007  
siehe auch Ogden, C., Richards, I.; The Meaning of meaning, 1923

## Modell

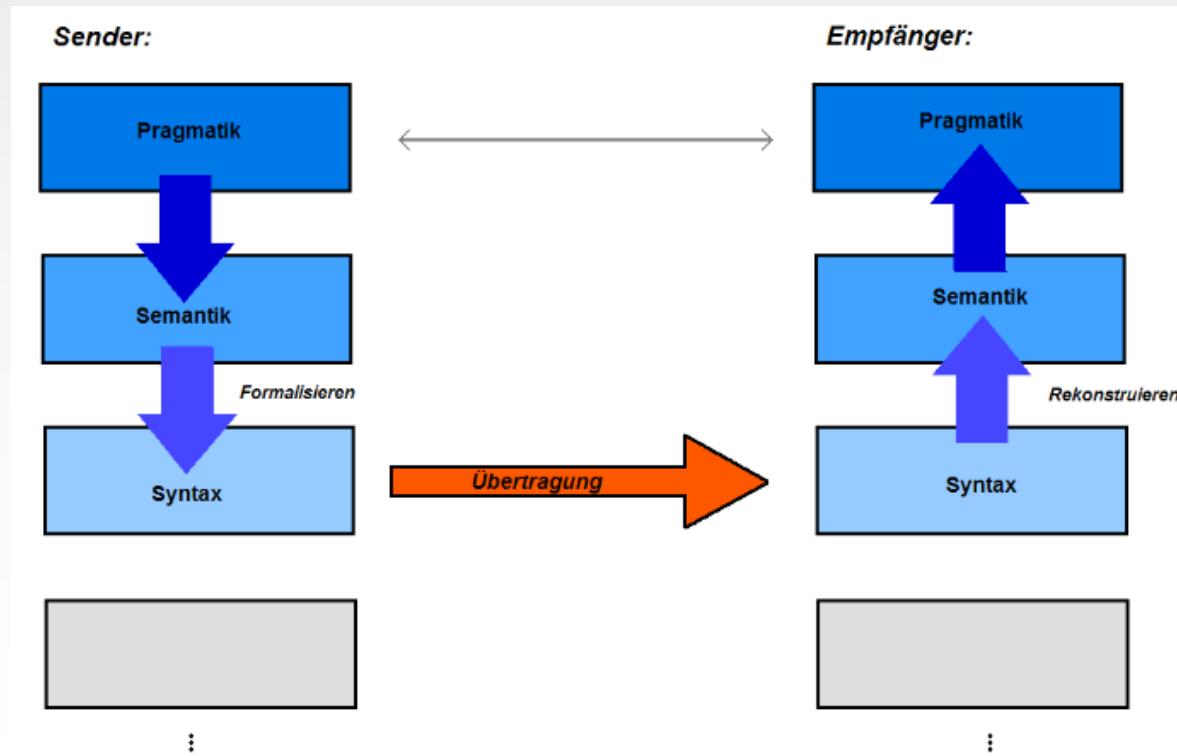


Quelle: (nach) Hesse, W.; Modellierung dynamischer und adaptiver Systeme, WiSo 2014/15  
[https://www.pst\\_ifi.lmu.de/Lehre/wise-15-16/modas/modas2.pdf](https://www.pst_ifi.lmu.de/Lehre/wise-15-16/modas/modas2.pdf); Zugriff: 29.10.2018

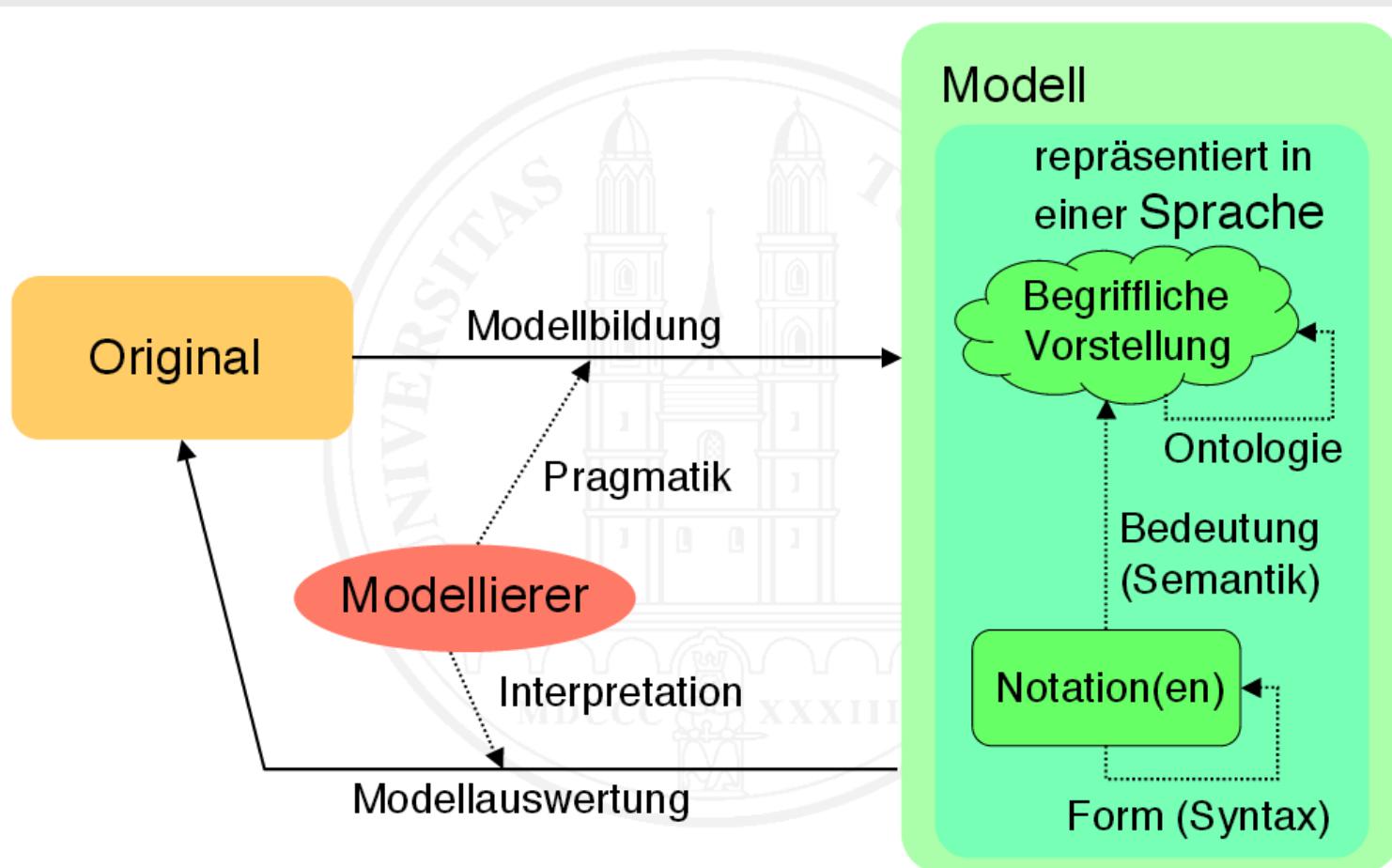


- Ein kognitives Modell beinhaltet Konzepte
- Eine Modell-Darstellung verwendet Symbole
- Ein Interpreter/Beobachter verwendet Modelle, die in einer Sprache ausgedrückt sind

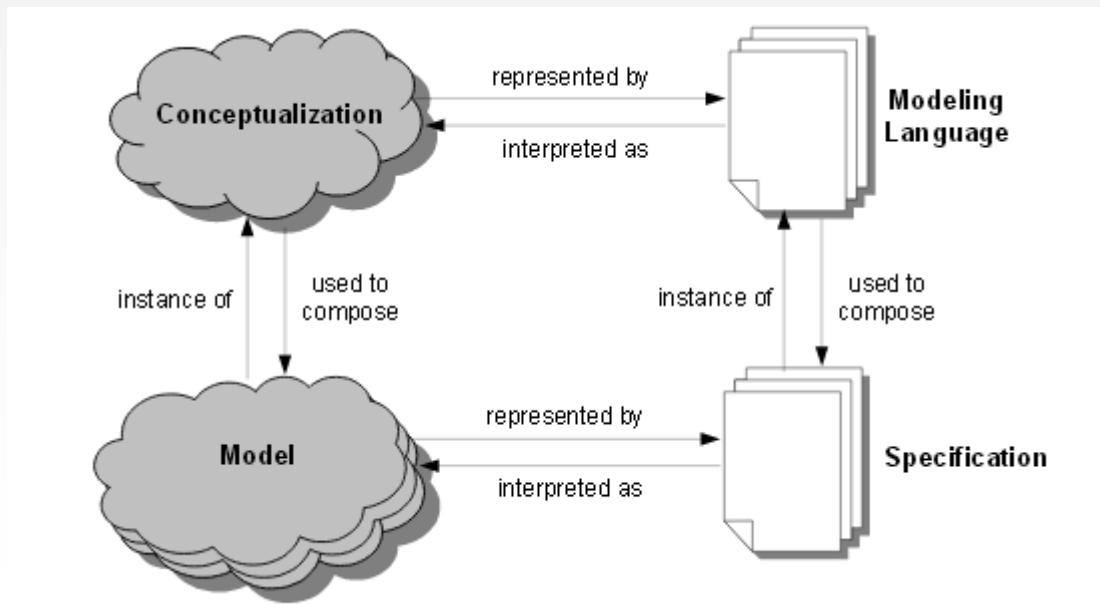
# Was? - Modell und Sprache (3)



Arnold, P: Information und Wissen, 2009  
<http://www.informatik.uni-leipzig.de/~graebel/Texte/Arnold-09.pdf>; Zugriff: 10.10.2018



„Conceptualizations and Models are abstract entities that only exist in the mind of the user or a community of users of a language. In order to be documented, communicated and analyzed they must be captured, i.e. represented in terms of some concrete artifact. This implies that a language is necessary for representing them in a concise, complete and unambiguous way.“



Guizzardi, G.; Ontological Foundations for Structural Conceptual Models, 2005  
<http://www.inf.ufes.br/~gguizzardi/OFSCM.pdf>; Zugriff: 24.11.2014

„Information is a function of the way that matter and energy are organized and arranged in space and time, a property of their *form* rather than their substance.“

Quelle: Reading; A.; Meaningful Information: The bridge between biology, brain, and behaviour, 2011

„It is not accidental that the word "form" appears in "information," since information is the amount of formal patterning ... in any system.“

Quelle: Miller, J.G., Living Systems - Basic concepts, 1978

## in-FORM-ation



$$\begin{array}{r} \text{8} \\ + \text{2} \\ \hline \text{10} \end{array}$$

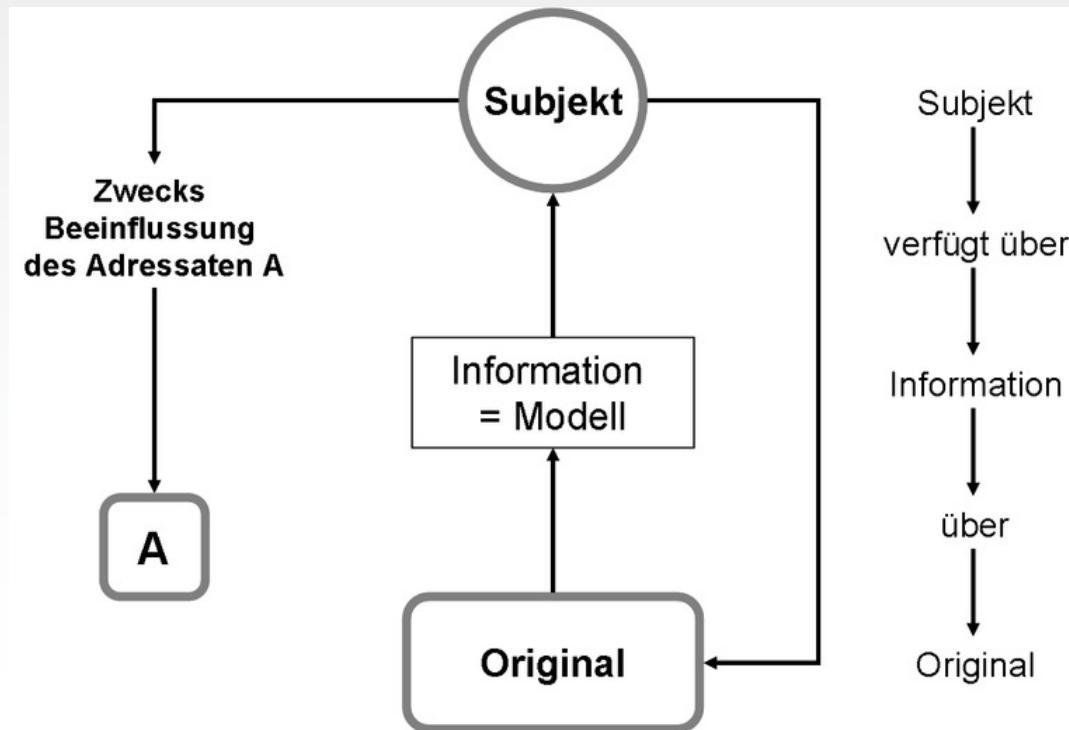
$$\begin{array}{r} \text{3} \\ + \text{5} \\ \hline \text{8} \end{array}$$

Quelle: Urs Wehrli, „Die Kunst Aufzuräumen“  
<http://antiecroton.com/wp-content/uploads/2013/07/philosophie.png> (Zugriff: 02.04.2014);  
<https://www.sueddeutsche.de/bildung/knobelei-der-woche-knacken-sie-das-streichholzraetsel-1.3117980>;  
Zugriff: 19.06.2022

# Was? - Informationsbegriff (2)

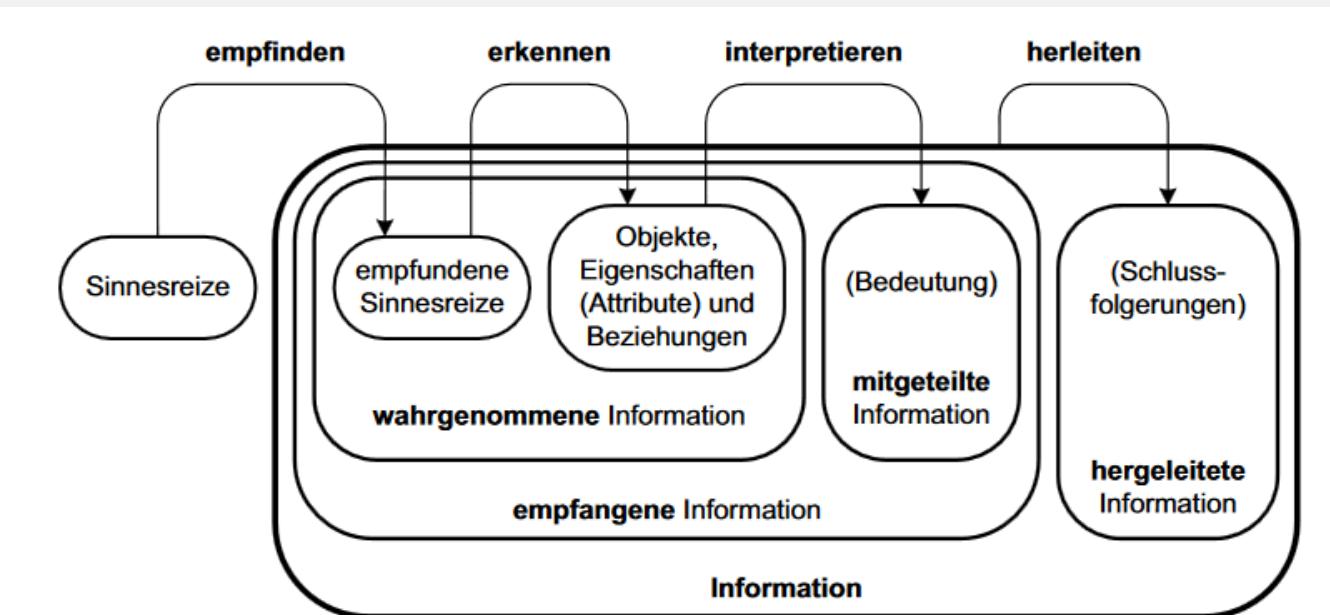
„Information als immaterielles Modell eines Originals für Zwecke eines Subjekts.“

Quelle: Steinmüller, W.: Eine sozialwissenschaftliche Konzeption der Informationswissenschaft. Nachrichten für Dokumentation, 1981

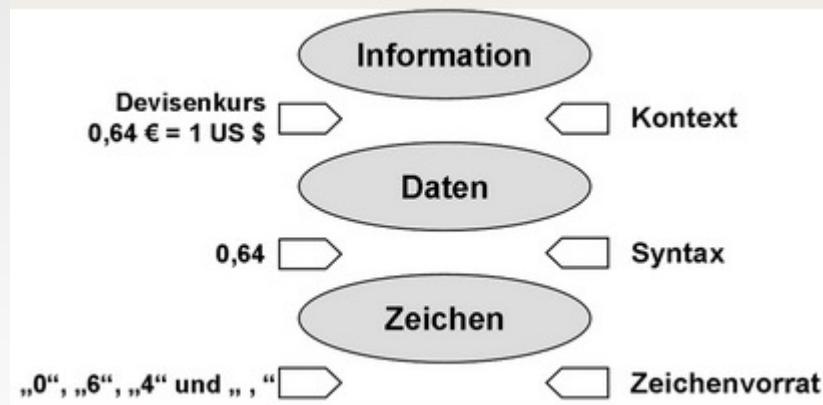


Quelle: Steinmüller, W.: Informationstechnologie und Gesellschaft: Einführung in die Angewandte Informatik, 1993

# Was? – Informationsbegriff (3)



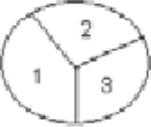
Quelle: Tabeling, P.; Softwaresysteme und ihre Modellierung, Springer 2006



„Auf der untersten Ebene befindet sich ein großer Vorrat verschiedener Zeichen als Basis aller weiter oben angesiedelten Begriffe. Werden die Zeichen einem Alphabet zugeordnet, kann man von Daten sprechen. Die Anreicherung mit zusätzlichem Kontext verschafft den Daten Bedeutung, so dass Information entsteht, bspw. darüber, dass mit 0,64 der Wert des Dollars in EURO gemeint ist.“

Quelle: Krcmar, H.; Information; in: Enzyklopädie der Wirtschaftsinformatik – Online Lexikon;  
<http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/daten-wissen/Informationsmanagement/Information-/index.html>;  
Zugriff: 23.02.2015

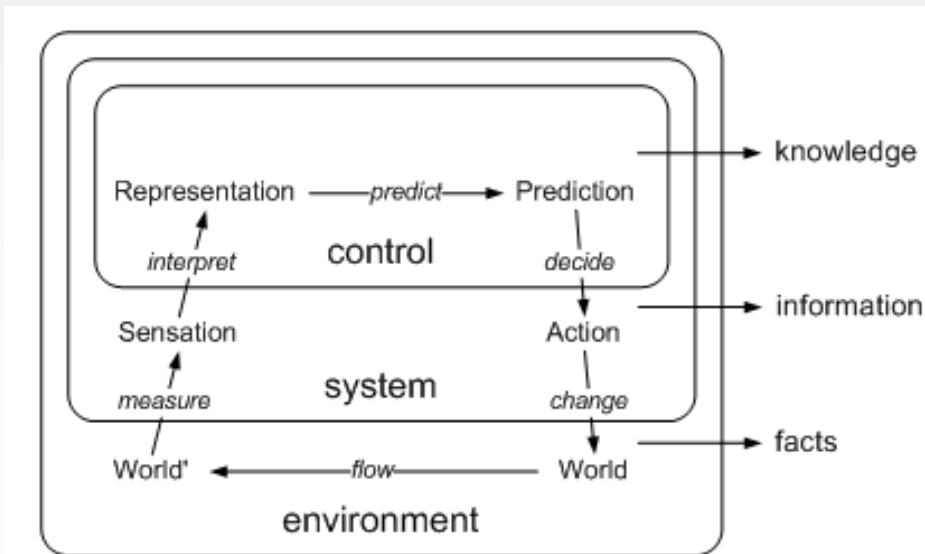
# Was? – Informationsbegriff (5)

Abgrenzung Beispiele	Zeichen	+ Syntax	+ Zweckbezug	+ Interpretation/ Klassifizierung
		= Daten	= Information	= Wissen
	0,1,2,3,... ABC... abc...	Umsatzliste: Nr. Stk. € 1 100 2000 2 80 1600 3 2 9,80 4 ...	Welche Kundengruppe kauft welche Produkte?	Warum kauft eine bestimmte Kundengruppe ein bestimmtes Produkt?  

Quelle: Einführung und Überblick Informationssysteme, FHNW;

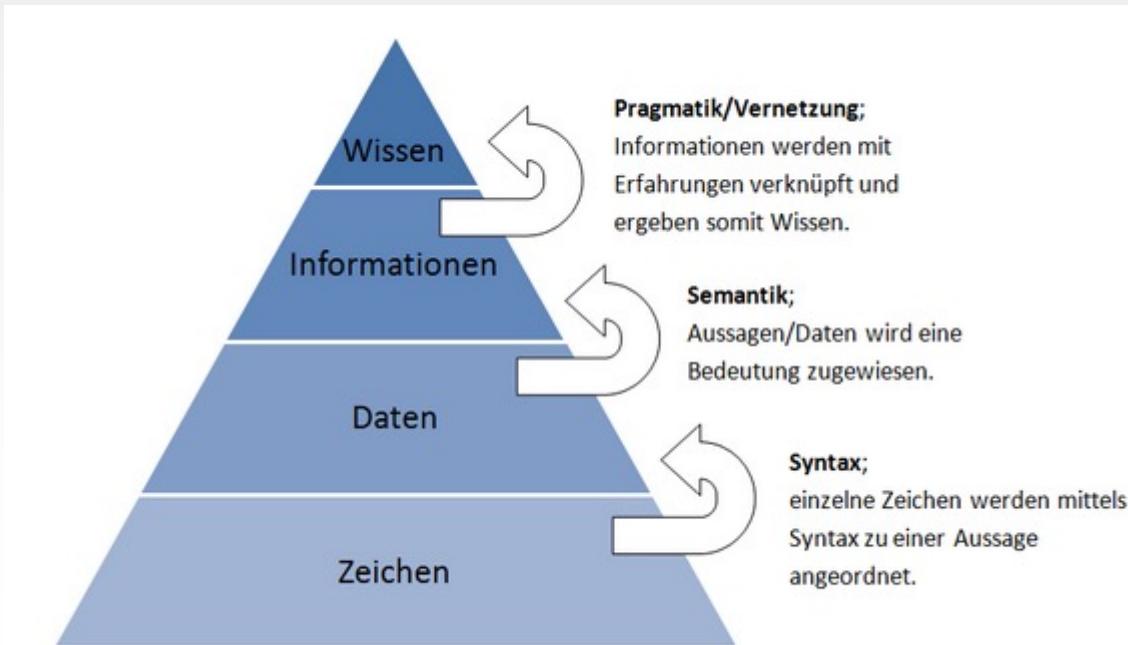
Zugriff: [https://web.fhnw.ch/plattformen/vrit/imo/ONLINE/05\\_IS/EinfuehrungIS.pdf](https://web.fhnw.ch/plattformen/vrit/imo/ONLINE/05_IS/EinfuehrungIS.pdf); 10.10.2018

# Was? - Informationsbegriff (6)



Quelle: Garcia, R.; A Semantic Web Approach to Digital Rights Management, 2007;  
<http://rhizomik.net/html/~roberto/thesis/html/KnowledgeRepresentation.html>; Zugriff: 05.11.2019

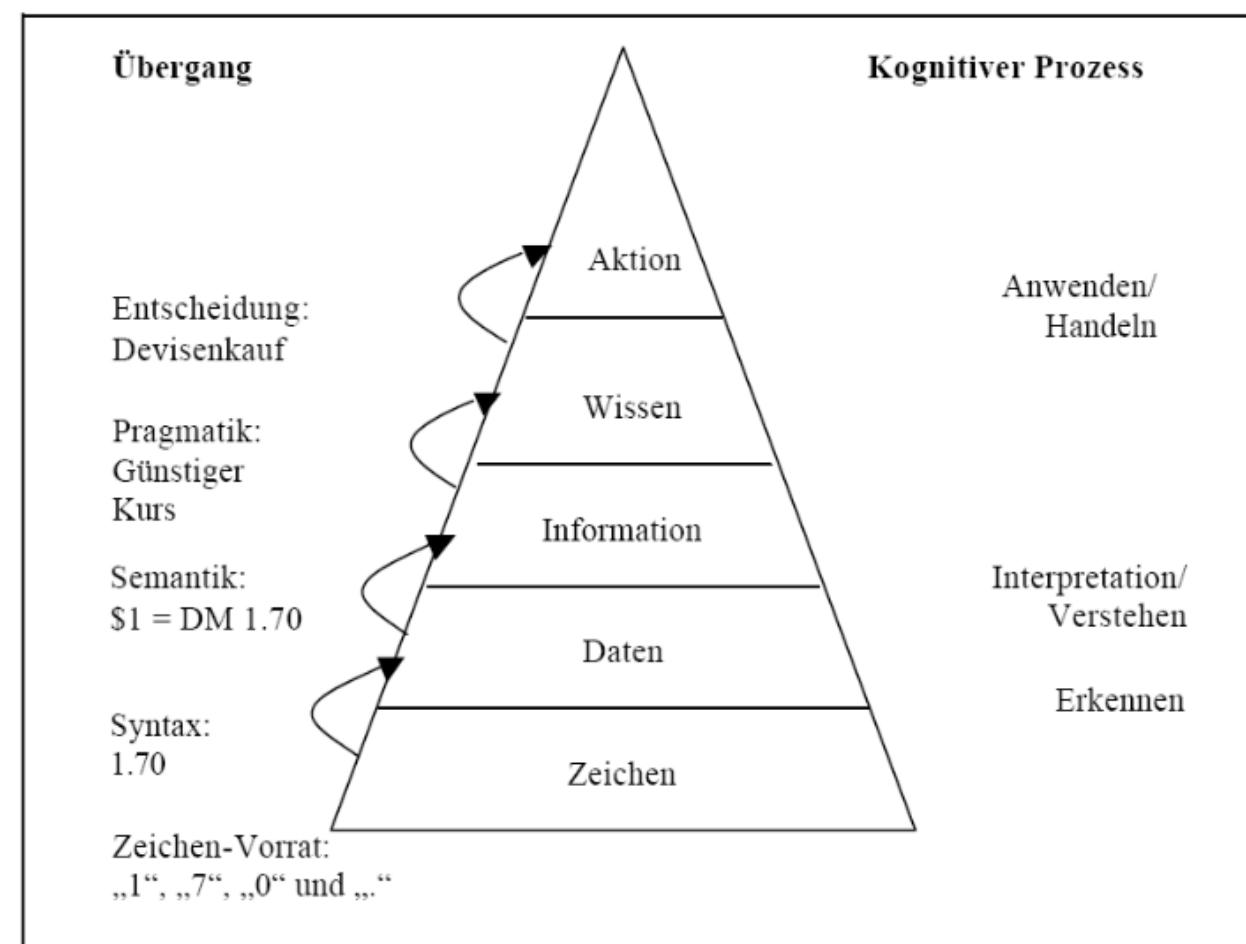
## Informationsbegriff und Sprache



Quelle: Hermann, R.; Wissenspyramide;

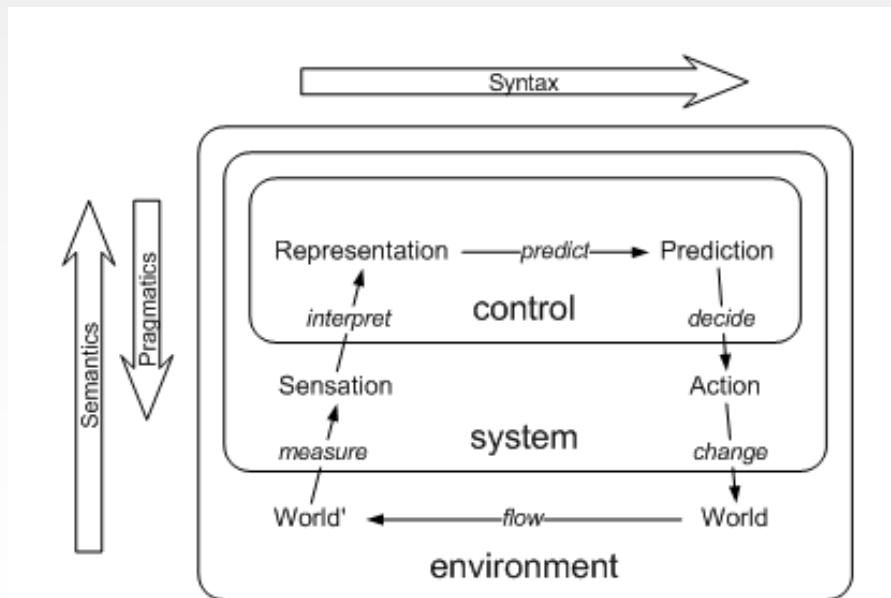
Zugriff: <https://derwirtschaftsinformatiker.de/2012/09/12/it-management/wissenspyramide-wiki/>; 10.10.2018

## Informationsbegriff und Sprache



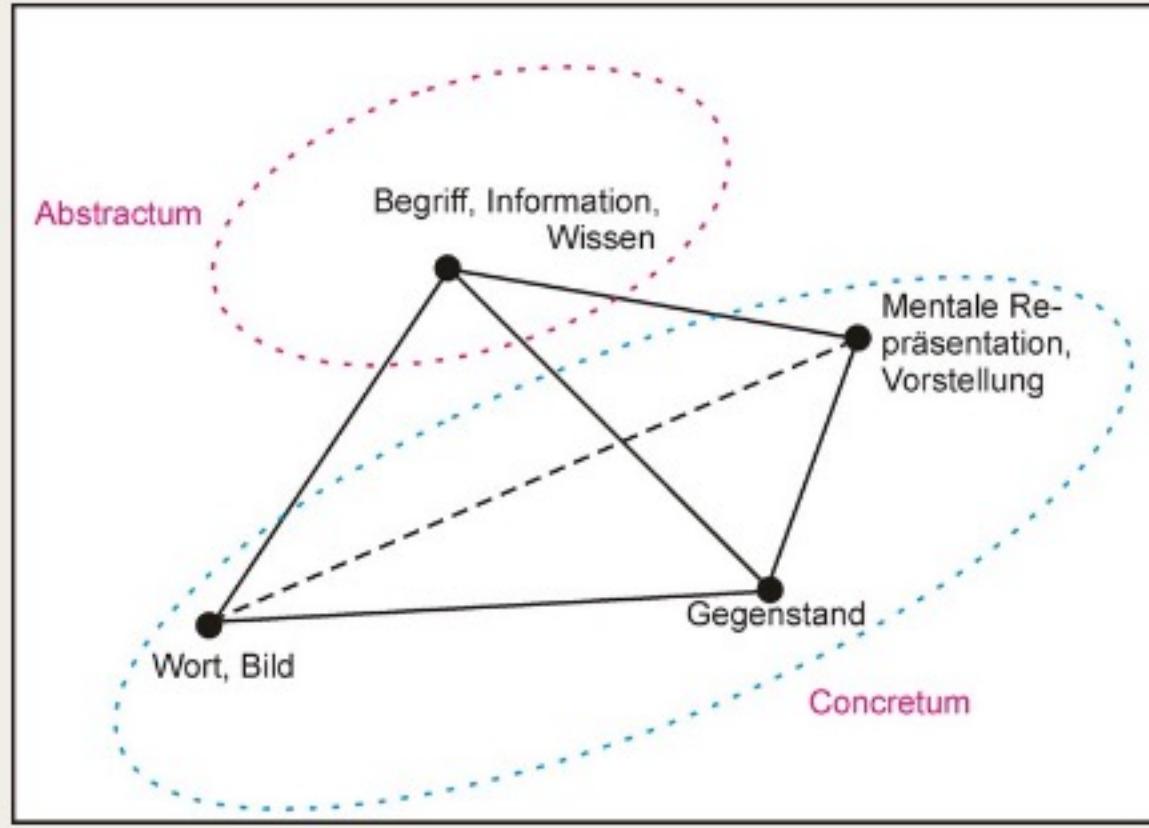
Quelle: Fuchs-Kittowski, K.; Wissens-Ko-Produktion – Verarbeitung, Verteilung und Entstehung von Informationen in kreativ lernenden Organisationen, 2000  
[http://www.wissenschaftsforschung.de/JB00\\_9-88.pdf](http://www.wissenschaftsforschung.de/JB00_9-88.pdf); Zugriff: 17.06.2019

## Informationsbegriff und Sprache



Quelle: Garcia, R.; A Semantic Web Approach to Digital Rights Management, 2007;  
<http://rhizomik.net/html/~roberto/thesis/html/KnowledgeRepresentation.html>; Zugriff: 05.11.2019

## Informationsbegriff und Modell



Quelle: Ortner, E.; Informationsbegriff in der Semiotik

[http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/daten-wissen/  
Informationsmanagement/Information-/Informationsbegriff-in-der-Semiotik](http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/daten-wissen/Informationsmanagement/Information-/Informationsbegriff-in-der-Semiotik); Zugriff: 13.11.2013

## Informationssystem und Modell

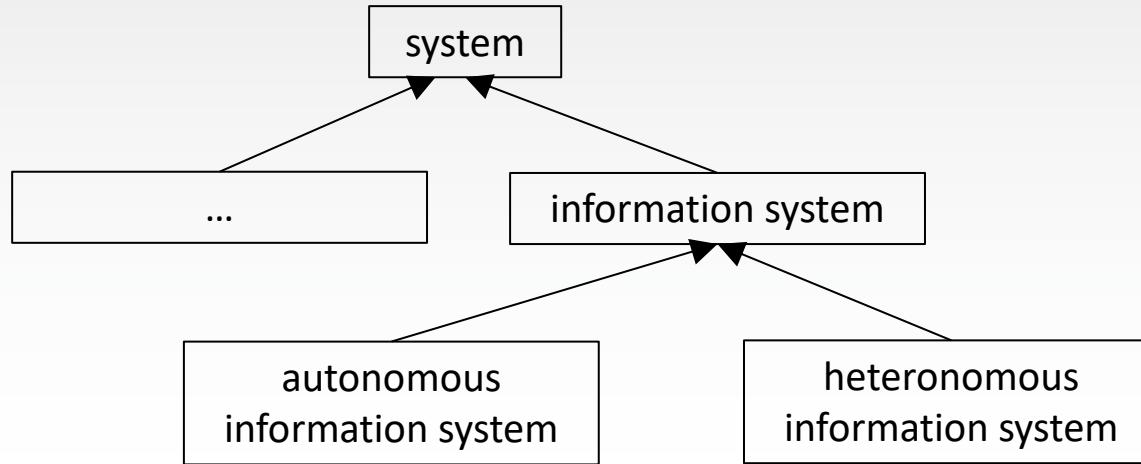
„Indeed, an information system can be viewed as a representation, or a model, of another system (usually termed the real system).“

Quelle: Wand, Y.; Monarchi, D., Parsons, J.; Woo, C.; Theoretical foundations for conceptual modeling in information systems development, Decision Support Systems 15, 1995

„... in their essence, computerized information systems are representations of real world systems.“

Quelle: Krogstie, J., Model-Based Development and Evolution of Information Systems: A Quality Approach, Springer 2012

## Informationsbegriff und System



### Further important characteristics

mobile	stationary
artificial	natural
virtual	physical
living	non-living
adaptive	fixed
distributed	co-located

„Ein Informationssystem (IS) ist im engeren Sinne (und so wird es i.d.R. verstanden) ein computergestütztes Anwendungssystem, d.h. ein Softwaresystem zur Ausführung betrieblicher Aufgaben. Im weiteren Sinne werden die Technik (Hard- und Software), die Menschen und die Anwendungen in einem Informationssystem zusammengefasst, das auch als Informations- und Kommunikationssystem (IuK-System) bezeichnet wird.“

Quelle: Gabriel, R.; Informationssystem; Enzyklopädie der Wirtschaftsinformatik;

<http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/uebergreifendes/Kontext-und-Grundlagen/Informationssystem>  
(Zugriff: 02.04.2014)

„... in information systems engineering, we should restrict the definition [of information systems] to designed systems, that is, systems that are designed and built by an engineer. The restriction is necessary because there are natural systems that perform information-processing functions ... in cognitive science, the human mind is viewed as a complex system that receives, stores, processes, and distributes information.

... an information system is a designed system that collects, stores, processes, and distributes information about the state of a domain.

A [information] system is therefore considered to have three main functions (Fig. 1.1):

1. *Memory*: to maintain a representation of the state of a domain.
2. *Informative*: to provide information about the state of a domain.
3. *Active*: to perform actions that change the state of a domain. “

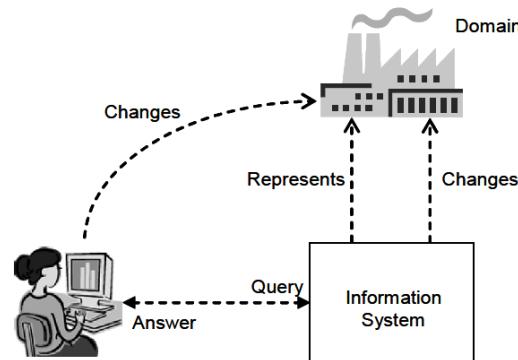
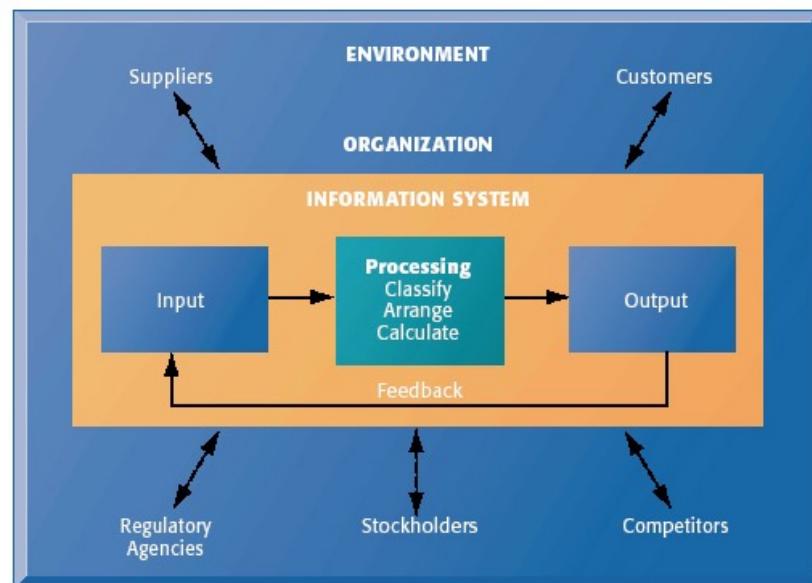


Fig. 1.1. Functions of an information system

Quelle: Antoni, O.; Conceptual Modeling of Information Systems, 2007

„An information system can be defined technically as a set of interrelated components that collect (or retrieve), process, store, and distribute information to support decision making and control in an organization.“

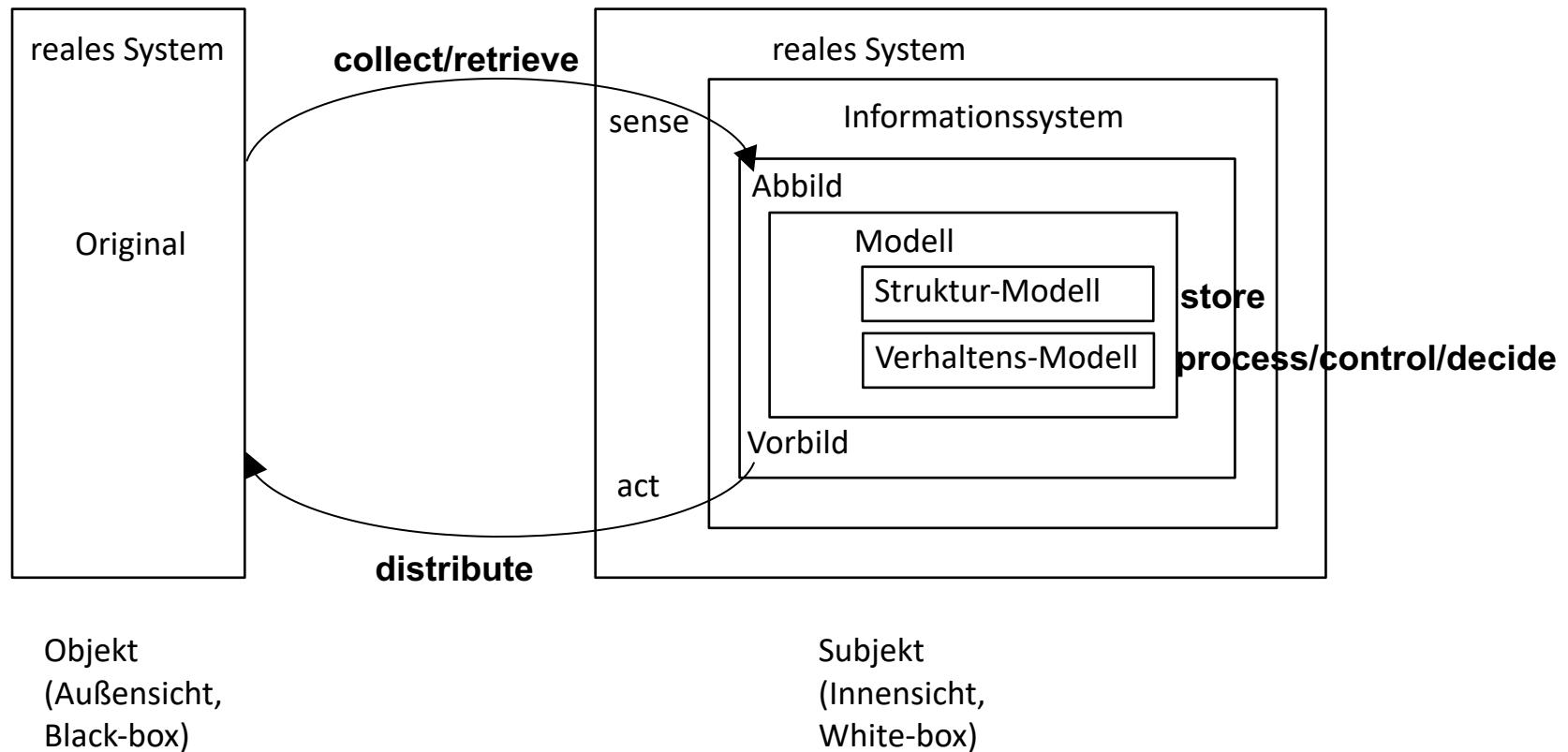


Quelle: Laudon, K.; Laudon, J.; Management Information Systems: Managing the Digital Firm; 2017

# Was? – Informationssystem (5)

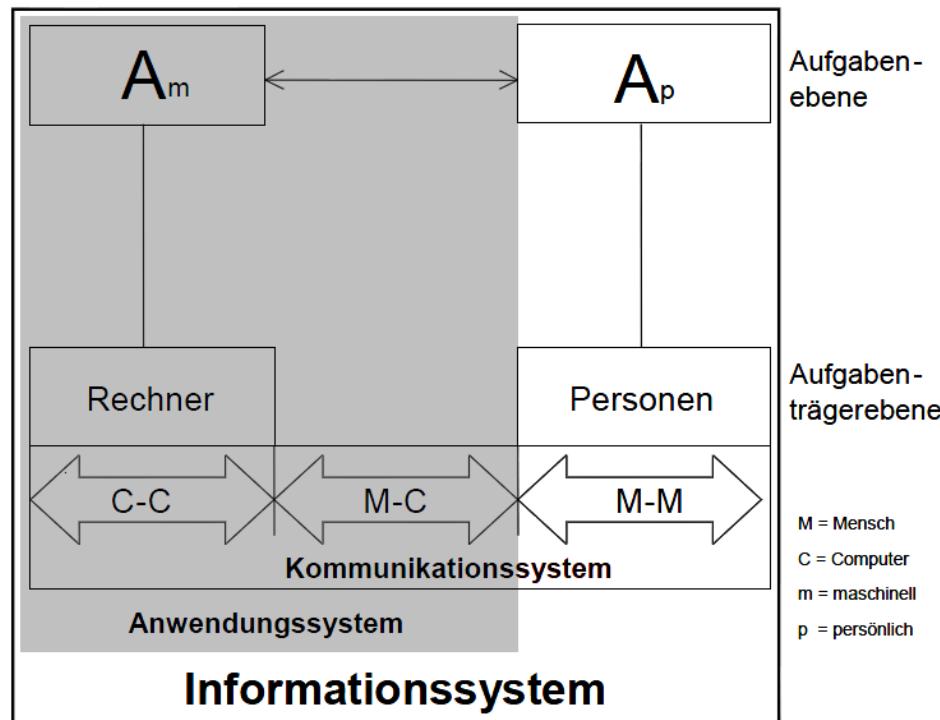
„An information system can be defined technically as a set of interrelated components that collect (or retrieve), process, store, and distribute information to support decision making and control in an organization.“

Quelle: Laudon, K.; Laudon, J.; Management Information Systems: Managing the Digital Firm; 2017



“Ein Informationssystem ... besteht aus Menschen und Maschinen ..., die Informationen erzeugen und / oder benutzen und die durch Kommunikationsbeziehungen miteinander verbunden sind.“

Quelle: Hansen H. et al.; Wirtschaftsinformatik, 11. Aufl.; 2015



Quelle: Einführung und Überblick Informationssysteme, FHNW;  
Zugriff: [https://web.fhnw.ch/plattformen/vrit/imo/ONLINE/05\\_IS/EinfuehrungIS.pdf](https://web.fhnw.ch/plattformen/vrit/imo/ONLINE/05_IS/EinfuehrungIS.pdf); 10.10.2018

## Informationssystem - Struktur und Verhalten

„If an information system is intended to be a representation of a real-world system, the grammars used during the design and implementation process must be capable of fully describing the structure (statics) and behavior (dynamics) of the real world.“

Quelle: Wand, Y.; Weber, R.; Toward a Theory of the Deep Structure of Information Systems; 1990

## Informationssystem – autonomous vs heteronomous

„an autonomous system is a self-determining system, as distinguished from a system determined from the outside, or a heteronomous system.“

Quelle: Thompson, E.; Mind in life; 2007

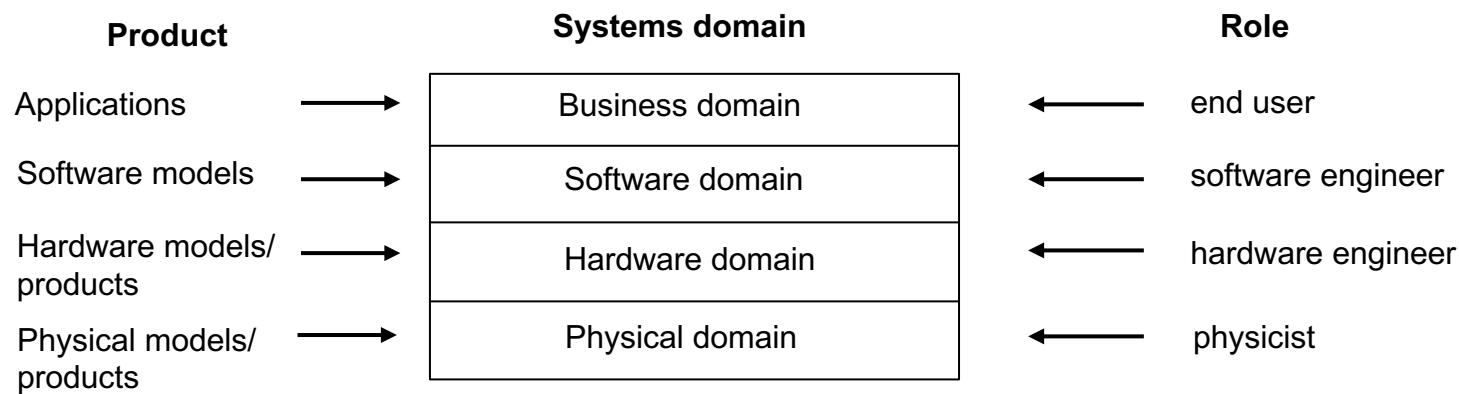
## Application

„... conceptual modelling languages are intended specifically for describing the application domain. The meaning of constructs used in these languages should be defined in terms of application domain concepts “

Quelle: Evermann, J.; Wand, Y.; Ontology based object-oriented domain modelling: fundament concepts, 2005

„To distinguish between the various domains, we will use the term *application domain* for customer-level domains ... A software system is typically associated with an application domain. We will use the term *software domain* to refer to domains that are identified by software engineers as such.“

Quelle: Bosch, J.; Design & Use of Software Architecture, 2000



Layered / partitioned software architecture

# Womit?

明白我

明白我  
Míngbái wǒ

„Die Grenzen meiner Sprache sind die Grenzen meiner Welt.“

Ludwig Wittgenstein

„... die Systeme von Einheiten und Regeln, die den Mitgliedern von Sprachgemeinschaften als Mittel der Verständigung dienen“

Quelle: <http://de.wikipedia.org/wiki/Sprache>; Zugriff: 27.04.2015

„Language matters to us because it is a vehicle for meaning – it allows us to take the desires, intentions, and experiences in our heads and transmit a signal through space that makes those thoughts pop up in someone else's head.“

Quelle: Bergen, B., Louder than Words: The new Science of How the Mind Makes Meaning; 2012

Sprachen können in **natürliche und künstliche Sprachen** unterteilt werden. Unter einer natürlichen Sprache versteht man eine vom Menschen „gesprochene“ Sprache in seiner Alltagswelt, die er als „Muttersprache“ erlernt hat (z.B. deutsch, englisch). Als künstliche Sprache versteht man eine Sprache, die gezielt für einen Zweck entwickelt wurde (z.B. Programmiersprachen (z.B. Java) bzw. Analyse- und Entwurfssprachen (z.B. UML)).

Sprache dient dazu die komplexe „Welt“ in handhabbaren Einheiten (Informationen) zu fassen und **Vorstellungen** über die „Welt“ durch **Kommunikation** mit anderen Menschen/Maschinen zu vermitteln.

Da die „Welt“ raum-zeitlich gegliedert erscheint, gibt es in Sprachen auch „**Begriffe**“ für **räumliche Strukturen und zeitliches Verhalten**.

Sprachen weisen generell die 4 aufeinander aufbauenden Ebenen **Lexikalik, Syntax, Semantik** und **Pragmatik** auf. Die gültigen Zeichen einer Sprache werden in der Lexikalik definiert. In der Syntax werden die Zeichen und Beziehungen (Regeln) angegeben. Die Bedeutung der Zeichen bestimmt die Semantik. Welchen Nutzen einzelne Sätze einer Sprache für die Kommunikationspartner haben, wird in der Pragmatik untersucht.

„How do languages work? The simple answer is that languages work *through representation*. They are 'systems of representation' ... because they all use some element to stand for or represent what we want to say, to express or communicate a thought, concept, idea or feeling.

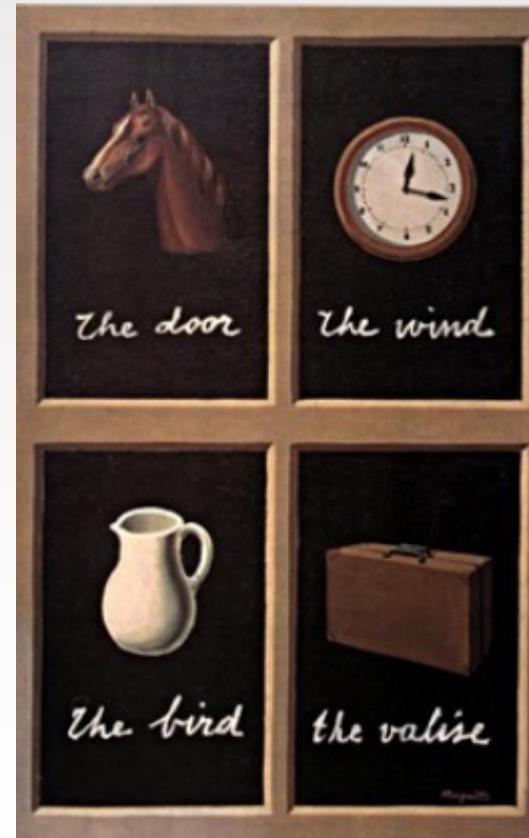
... but their [- the elements of a language - ] importance for a language is not what they *are* but what they *do*, their function. They construct meaning and transmit it. They signify. They don't have any clear meaning *in themselves*. Rather, they are the vehicles or media which *carry meaning* because they operate as *symbols*, which stand for or represent (i.e. symbolize) the meanings we wish to communicate. To use another metaphor, they function as *signs*. Signs stand for or represent our concepts, ideas and feelings in such a way as to enable others to 'read', decode or interpret their meaning in roughly the same way we do.“

Hall, R.; Representation: Cultural Representation and Signifying Practices, 2003

## Representation - Symbol

„Language is **arbitrary**, which means that verbal symbols are not intrinsically connected to what they represent.“

Quelle: Wood, J.; Communication Mosaics: An Introduction to the Field of Communication; 8<sup>th</sup> ed., 2016



Quelle: Magritte, R.; The Key of dreams, 1930

„Die Betrachtung der Sprache erfolgt unter zwei Hauptaspekten:

- unter funktionalen Aspekten = Sprache als **Kommunikationsmittel**  
(Kommunikationswissenschaft)
- unter formalem Aspekt = Sprache als **Zeichensystem**  
(Semiotik)“

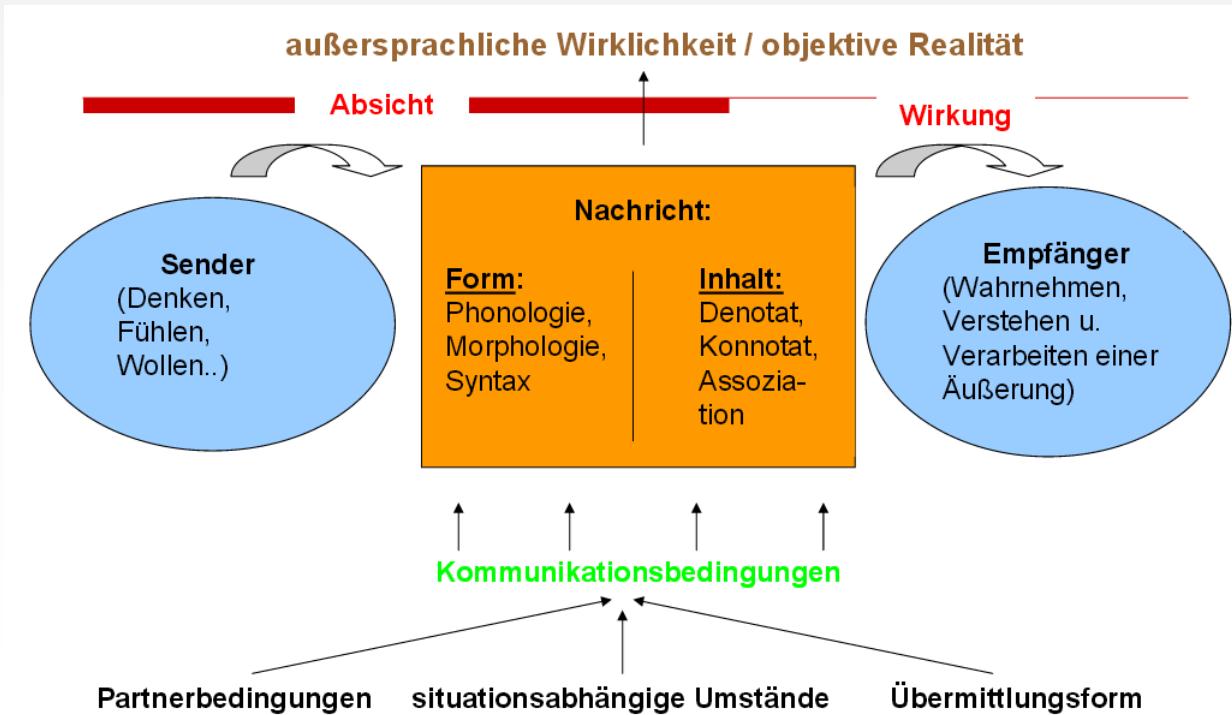
Quelle: Iakushevich, M.; Einführung in die Germ. Sprachwissenschaft, WS 09/10  
[http://kw.uni-paderborn.de/fileadmin/kw/institute-einrichtungen/germanistik-und-vergleichende-literaturwissenschaft/germanistik/Personal/Iakushevich/2.Sitzung\\_26.10.pdf](http://kw.uni-paderborn.de/fileadmin/kw/institute-einrichtungen/germanistik-und-vergleichende-literaturwissenschaft/germanistik/Personal/Iakushevich/2.Sitzung_26.10.pdf) (Zugriff: 28.10.2014)

Linguistics = Communication + Symbolization/Signification

## Kommunikation

„Kommunikation ist die menschliche und im weitesten Sinne technisch fundierte Tätigkeit des wechselseitigen Zeichengebrauchs und der wechselseitig adäquaten Zeichendeutung zum Zweck der erfolgreichen Verständigung, Handlungskoordinierung und Wirklichkeitsgestaltung.“

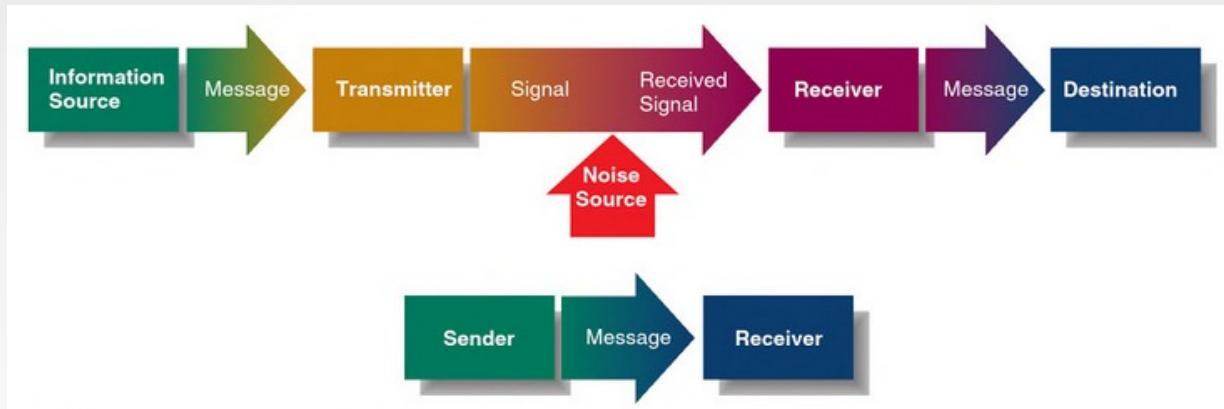
Quelle: Krallmann, D., Ziemann, A.; Grundkurs Kommunikationswissenschaft, UTB, 2001



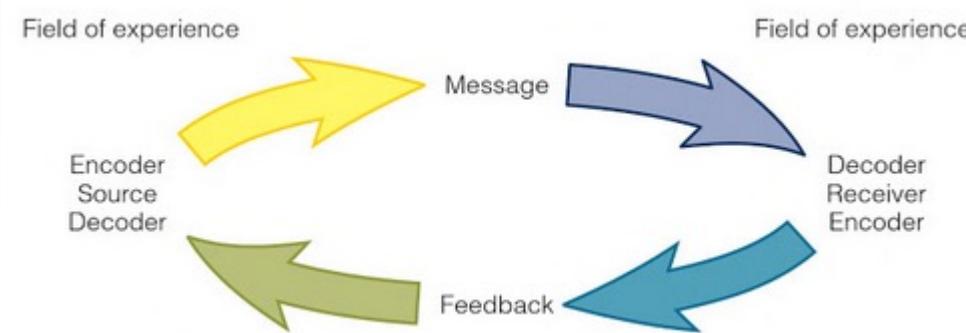
Quelle: Iakushevich, M.; Einführung in die Germ. Sprachwissenschaft, WS 09/10  
[http://kw.uni-paderborn.de/fileadmin/kw/institute-einrichtungen/germanistik-und-vergleichende-literaturwissenschaft/germanistik/Personal/Iakushevich/2.Sitzung\\_26.10.pdf](http://kw.uni-paderborn.de/fileadmin/kw/institute-einrichtungen/germanistik-und-vergleichende-literaturwissenschaft/germanistik/Personal/Iakushevich/2.Sitzung_26.10.pdf) (Zugriff: 28.10.2014)

## Kommunikation

### Linear Model

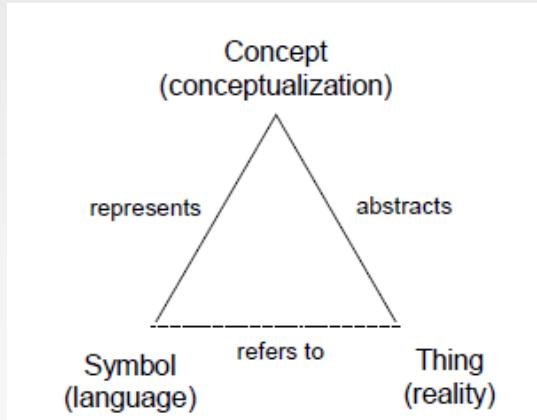


### Interactive Model

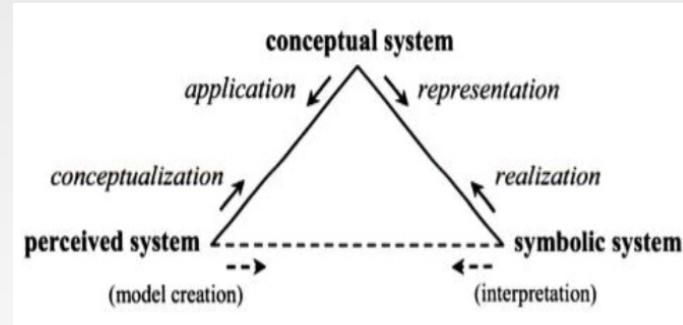


Quelle: Wood, J.; Communication Mosaics: An Introduction to the Field of Communication; 8<sup>th</sup> ed., 2016

## Semiotisches Dreieck oder Ullmann's Triangle bzw. modeling triangle



(a)

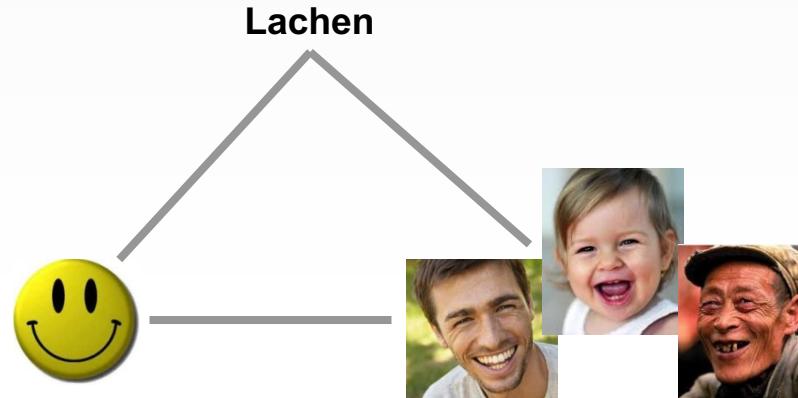


(b)

Quelle: (a) Guizzardi, G.; On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models, 2007

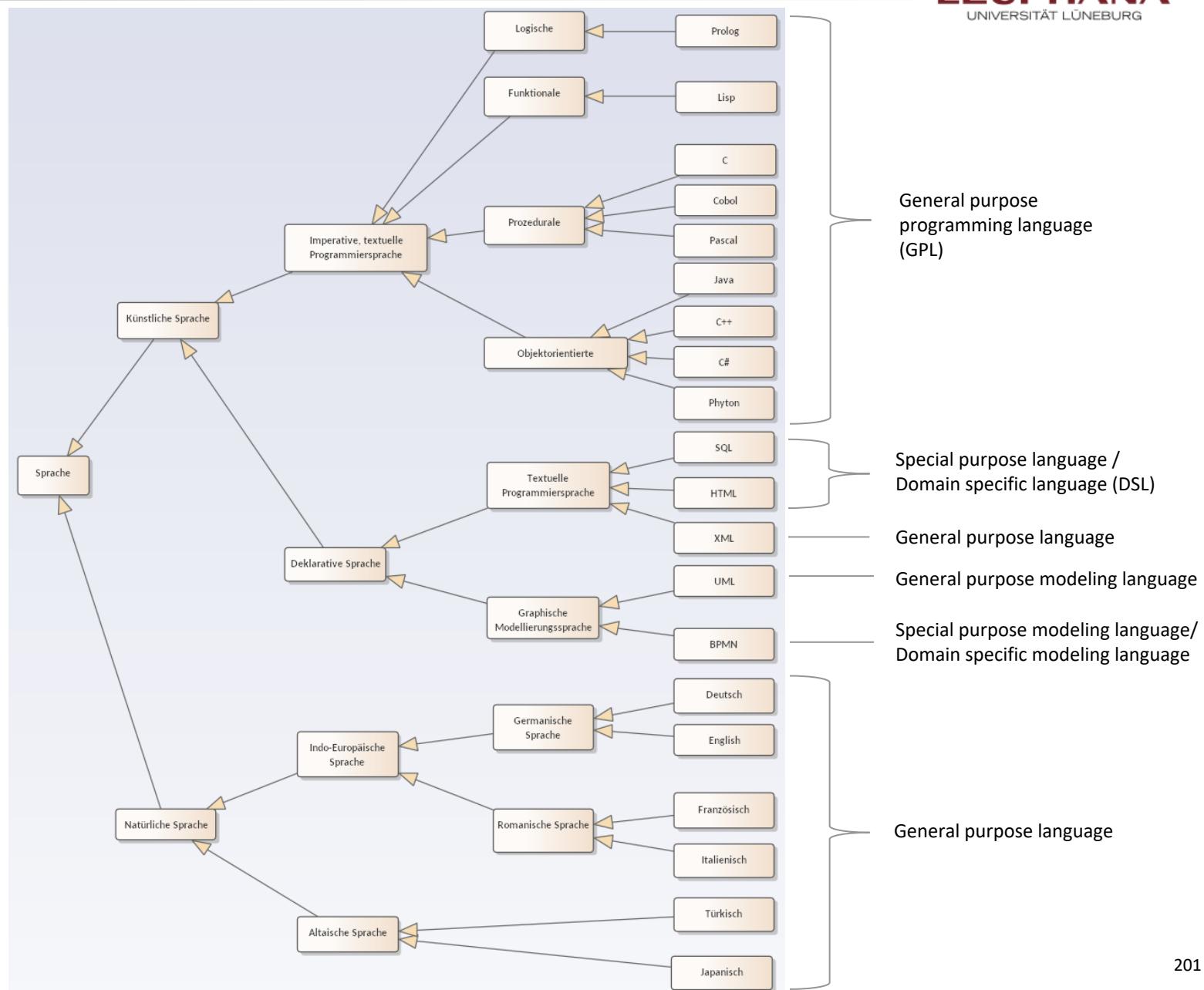
siehe auch Ogden, C., Richards, I.; The Meaning of meaning, 1923

(b) Sykes, J. A. (2003). Negotiating Early Reuse of Components—A Model-Based Analysis. *TEAM*, 66

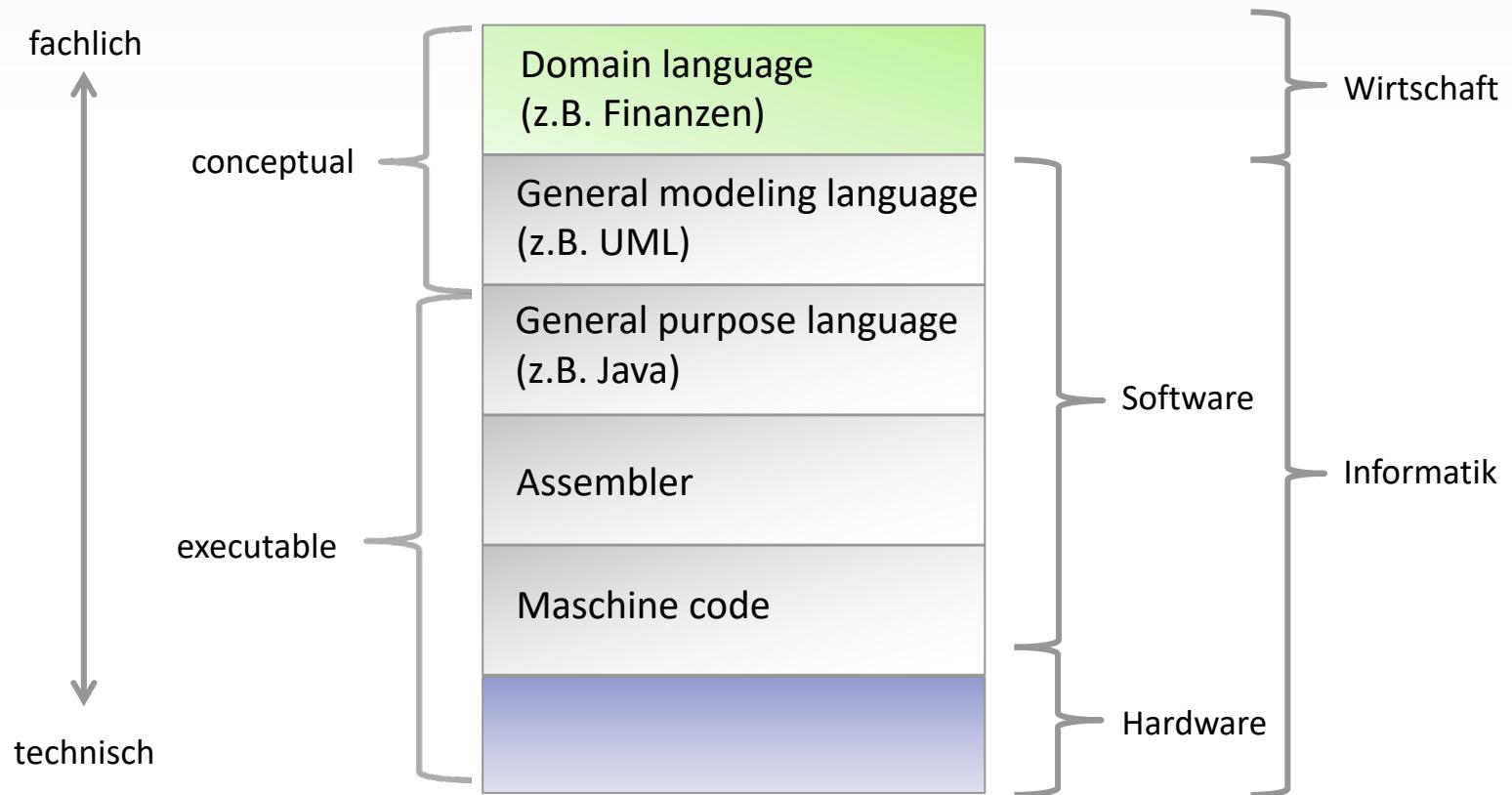


# Womit? – Sprache (9)

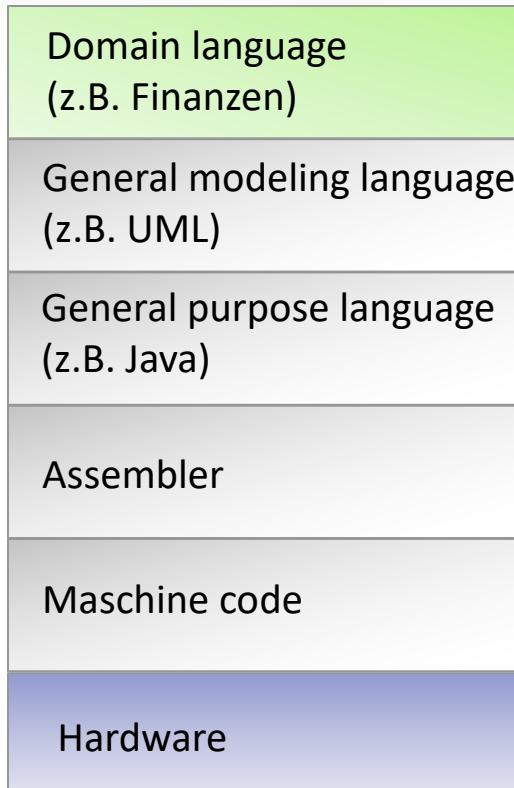
## Teilausschnitt der Sprachen-taxonomie



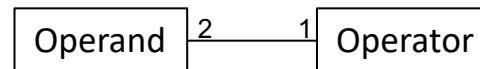
Im Bereich der Softwaretechnik bzw. Systemanalyse gibt es für unterschiedliche **Aufgaben/Rollen spezielle Sprachen**. So benutzt ein Softwarearchitekt eine Architektursprache (Architectural description language (ADL)), ein Systemanalytiker für die Definition von Anforderungen eine Fachsprache (Domain language), ein Systemdesigner eine Entwurfssprache (General modeling language, z.B. UML), ein Programmierer eine Programmiersprache (z.B. Java), ein Datenbankadministrator eine data definition language (DDL).



## Sprachschichten der Informatik



„Addiere x zu y“

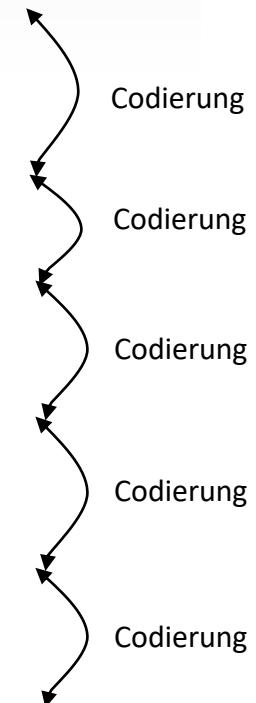


zahl1.add(zahl2);

ADD zahl1, zahl2

Operator Speicherstelle, Speicherstelle  
33 47,48

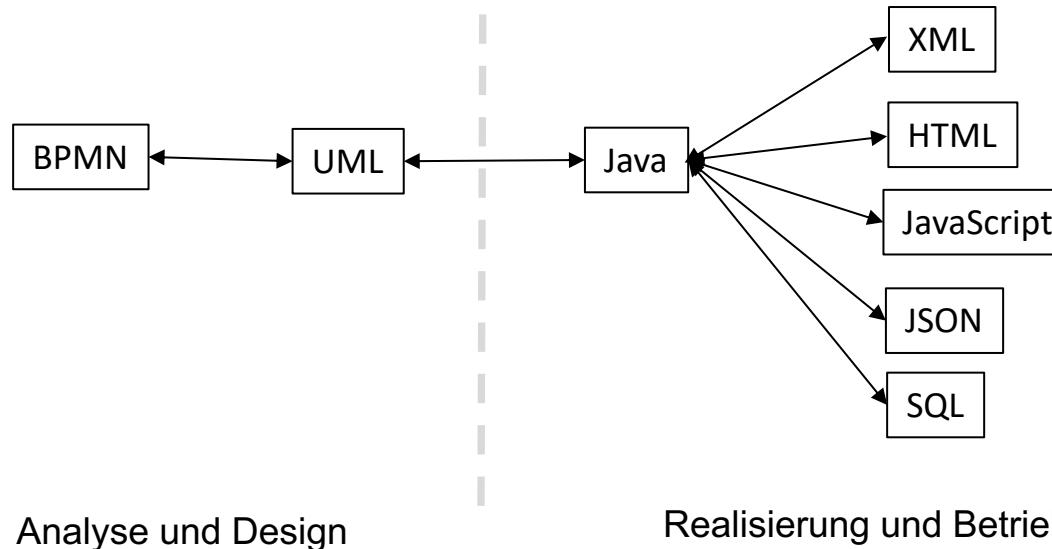
011000111101001000100



## Übersetzungen zwischen Sprachen der Informatik

Im Softwareentwicklungslebenszyklus werden zur Erstellung eines Softwaresystems mehrere Sprachen verwendet. Die Vielzahl an Sprachen ist durch die speziellen Aufgabe der Sprache bedingt (z.B. HTML zum Beschreiben der Seitenstruktur einer Web-Seite). Dennoch stehen einzelne Sprachen miteinander in Beziehung, so dass es einen Übersetzungsvorgang zwischen den Sprachen gibt. Die fachliche Anwendungsdomäne wird z.B. in BPMN/UML und später z.B. in Java beschrieben. Zusätzlich ist es aber notwendig, dass die fachlichen Strukturen und Abläufe für technische Endgeräte wie Browser oder rel. DBMS verfügbar gemacht werden. Es gibt also eine generelle Notwendigkeit zwischen Fachlichkeit und Technik zu übersetzen.

Teilmenge an Sprachen der Informatik und ihrer (Teil-)Übersetzung untereinander.



Analyse und Design

Realisierung und Betrieb

## Fachsprache

„Das Material wird durch Bandanlagen zu einem Bandsammelpunkt befördert und danach über einen Verschiebekopf zu einem Absetzer transportiert, der das Material in einen Bunker verstürzt. Alternativ kann der Abraum aus einer Sohle über einen Bandschleifenwagen auf eine Kippenstrosse verkippt werden.“

„Der Großbaum hängt gefahrlos über Kopfhöhe und wird über einen Traveller am Heck auf eine elektrische Zentralwindschottet.“

„Die Blätter sind mittels Kettenblattschrauben am Kurbelstern befestigt. Dieser ist zumeist 4-armig, gelegentlich 3-armig. ... Der Kurbelstern besitzt einen äußeren und einen inneren Lochkreis. ... Zur Demontage des mittleren Blattes muss der Kurbelarm abgenommen werden. Hierfür lösen Sie die Schraube an der Tretkurbel mit dem passenden Innensechskantschlüssel. Diese Schraube dient auch als Kurbelabzieher. Sie können den Arm nun vom Konus nehmen.“

„Der Schaftgrundschnitt eines Monkstraps entspricht dem eines Derbys, wobei das innere hintere Schaftteil, das so genannte *Quartier*, in einen oder zwei Riemen ausläuft, die mit einer bzw. zwei auf der Spitze des äußeren hinteren Schaftteils befindlichen Dornschließe geschlossen werden. ... Monks sind meist unverziert, doch gibt es sie auch als Halfbrogues und recht selten auch mit einer verzierten Flügelkappe.“

„Die Maische wird nun stufenweise erhitzt. Hierbei wird der Malzzucker vom Malzschrot gelöst. Die Spelzen werden beim Läutern von der Würze getrennt. Dies geschieht im Läuterbottich, wo die Würze durch den Siedeboden geleitet wird.“

„[Im] Kaliber bildet die Dreiviertelplatine das Gegenstück zur Werkplatte und überdeckt auf der Brückenseite drei Viertel des Werkes. Zwischen der Werkplatte und der Dreiviertelplatine sind das Räderwerk, Federhaus und Kronrad gelagert. Die Lagerungen der Unruh erfolgt allerdings in einem Kloben, der im Gegensatz zur Brücke eben nur an einer Stelle mit der Platine verschraubt ist. Am anderen Ende ragt er frei über die in ihm lagernde Unruh hinaus, manchmal auch über den Anker oder einzelne Räder. Mit einem Kloben wird die gangregulierende Unruh besonders inszeniert, weshalb zahlreiche Hersteller den Unruhkloben obendrein besonders schön finissieren – sei es mit feinen Satinierungen, gebrochenen Kanten, Genfer Streifen, ...“

## Fachsprache

"Am unteren Ende des Halms ist die Reite angebracht. Manchmal gibt es zwischen Halm und Reite eine Verdickung, die Gesenk genannt wird. Am oberen Ende des Halms ist der Bart, der an den Halm hart angelötet, mit ihm gegossen oder geschmiedet sein kann. Der Halm kann massiv (Volldorn) oder hohl (Hohldorn) ausgeführt sein."

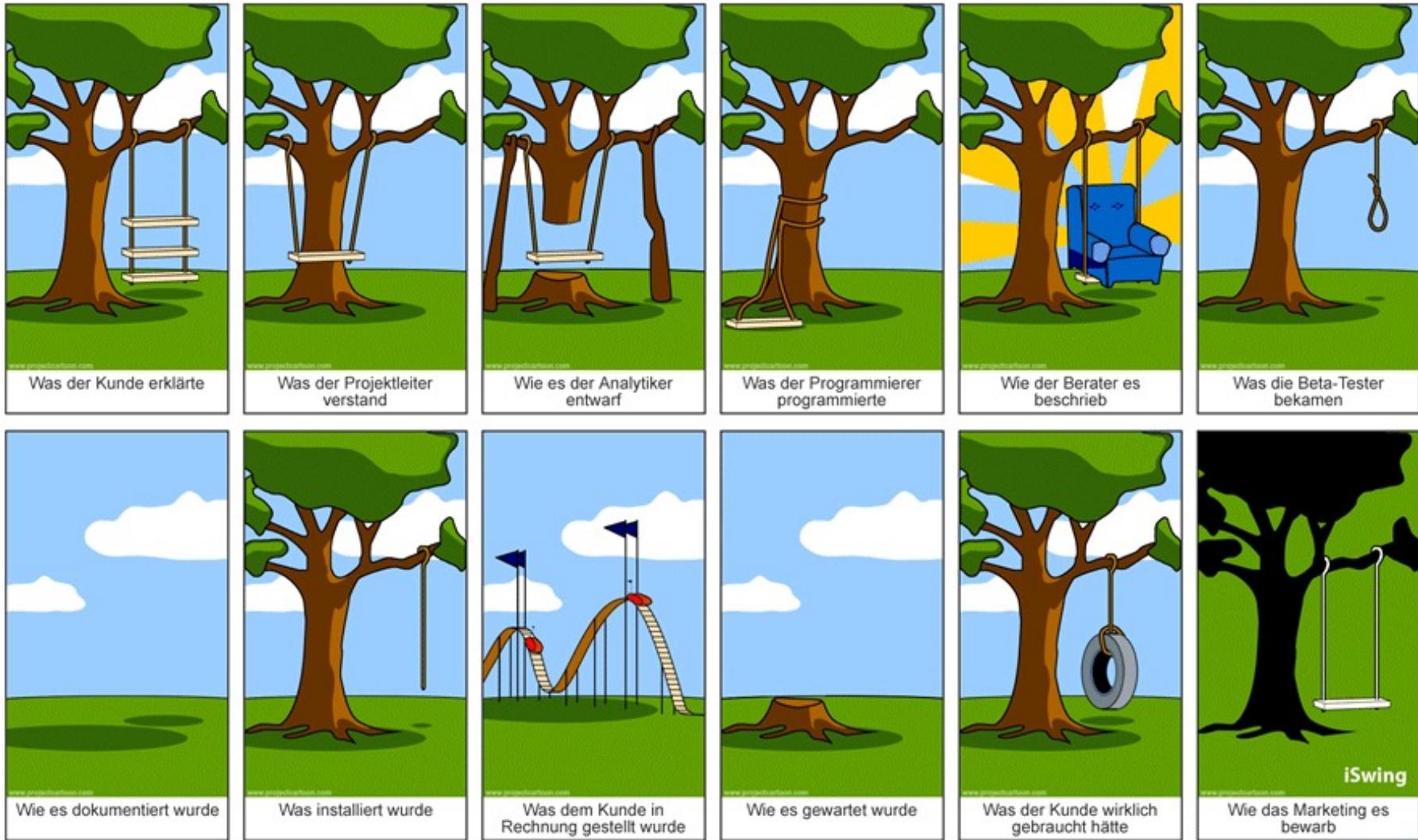
## Software engineering and communication

“Communication is the most critical and time-consuming activity in software engineering. Misunderstandings and omissions often lead to faults and delays that are expensive to correct later in the development. Communication includes the exchange of models and documents about the system and its application domain, reporting the status of work products, providing feedback on the quality of work products, raising and negotiating issues, and communicating decisions. Communication is made difficult by the diversity of participants’ backgrounds, by their geographic distribution, and by the volume, complexity, and evolution of the information exchanged.”

Quelle: Bruegge, B., Dutoit, A., Object-oriented Software Engineering, 3rd ed., 2010

# Womit? - Sprache (14)

## Kommunikation und (Fach-)Sprache



Der Technikbegriff hat mehrere Facetten. Erstens können darunter **alle vom Menschen gemachten Gegenstände** verstanden werden. Zweitens gilt eine **erlernte Fähigkeit**, die es ermöglicht etwas angemessen zu tun, auch als Technik. Drittens ist eine Form des **planmäßigen Vorgehens** darunter zu verstehen.

Somit weißt Technik einen **Produktcharakter** (gemachte Gegenstände) als auch einen **Prozesscharakter** (erlernte Fähigkeit, planmäßiges Vorgehen) auf. Produkt und Prozess gehen miteinander eine **komplementäre Beziehung** ein, d.h. es gibt kein Produkt ohne einen Prozess und umgekehrt.

Wesentliches Kennzeichen für die Erstellung von Produkten durch Menschen ist die Verwendung von **Werkzeugen**. Werkzeuge sind für einen speziellen **Zweck** konstruiert um von einer Person (Rolle) für eine Abbildung der „Welt“ und die Veränderung der „Welt“ zu dienen.

Im Bereich der Softwaretechnik bzw. der Systemanalyse gibt es je nach Aufgabe/Rolle spezielle Werkzeuge. So benutzen z.B. Systemanalytiker spezielle Editoren innerhalb eines **CASE-Tools** (Computer Aided Software Engineering).

Wesentliche künstliche Produkte (**Artefakte**) stellen im Bereich der Softwareentwicklung **Dokumente und Modelle** dar.

# Überblick W-Fragen

Womit?

Sprache

Werkzeug

Wie?

Prinzipien

Methoden

Verfahren

Was?

Anforderungen

Modell

Warum?

System

komplex

adaptiv

Wer?

Rolle

Stakeholder

Sichten

Wozu?

Verständigung

Abgrenzung

Definition

Erstellung

Wo?

Logischer Raum

Schichten

Wann?

Software development life cycle

Phasen

**Womit?**  
UML

**Wie?**

Was? Wie?  
Fachlich / technisch  
Schnittstellen

**Was?**

Analysemodelle  
Lasten- bzw. Pflichtenheft /  
Product Backlog

**Warum?**

Unterschiedliche Sprache  
Fachliche (technische) Anforderungen

**Wer?**

Systemanalyst

**Wozu?**

Anforderungsbeschreibung  
Vertragsgrundlage

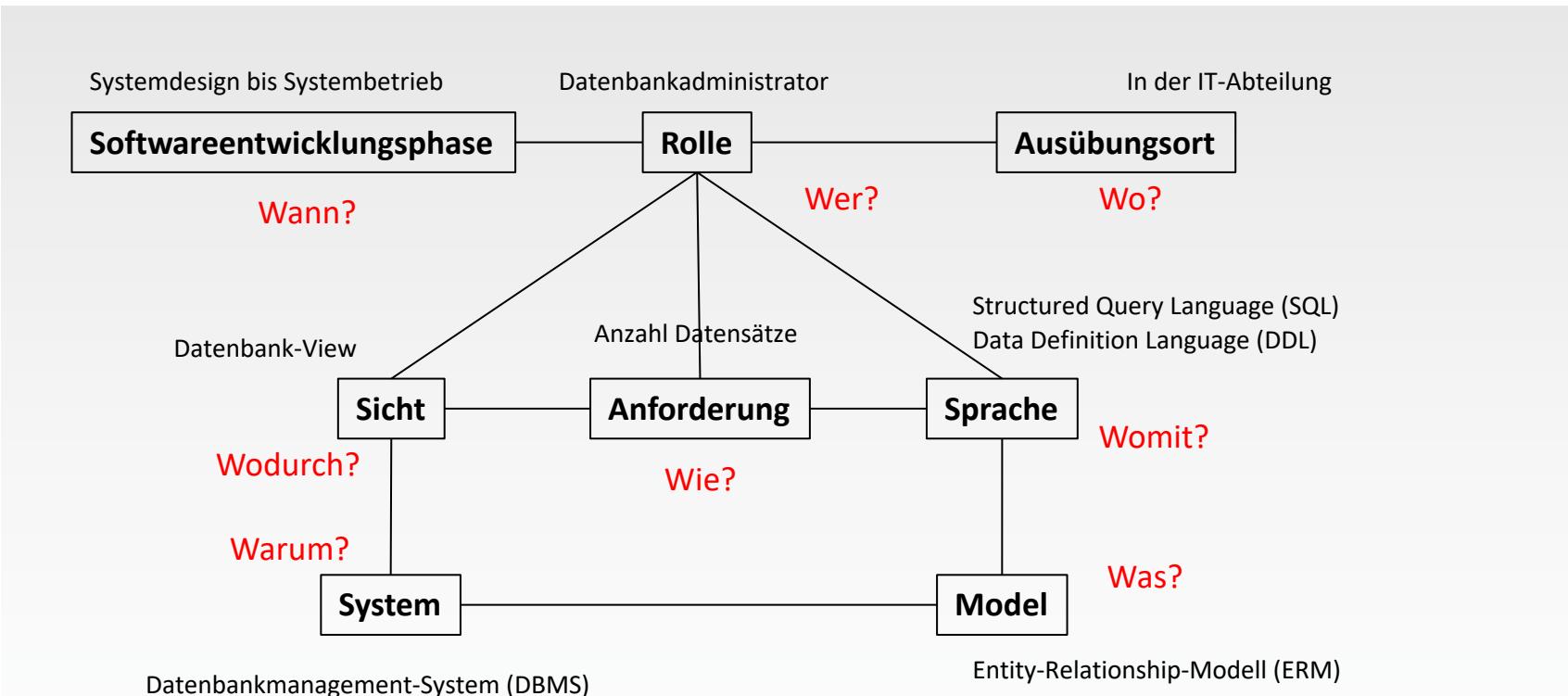
**Wo?**

Unterschiedliche fachliche funktionale  
Beziehungen (Heterarchie und Hierarchie)

**Wann?**

Analysephase

# Zusammenfassung W-Fragen (Beispiel Daten)



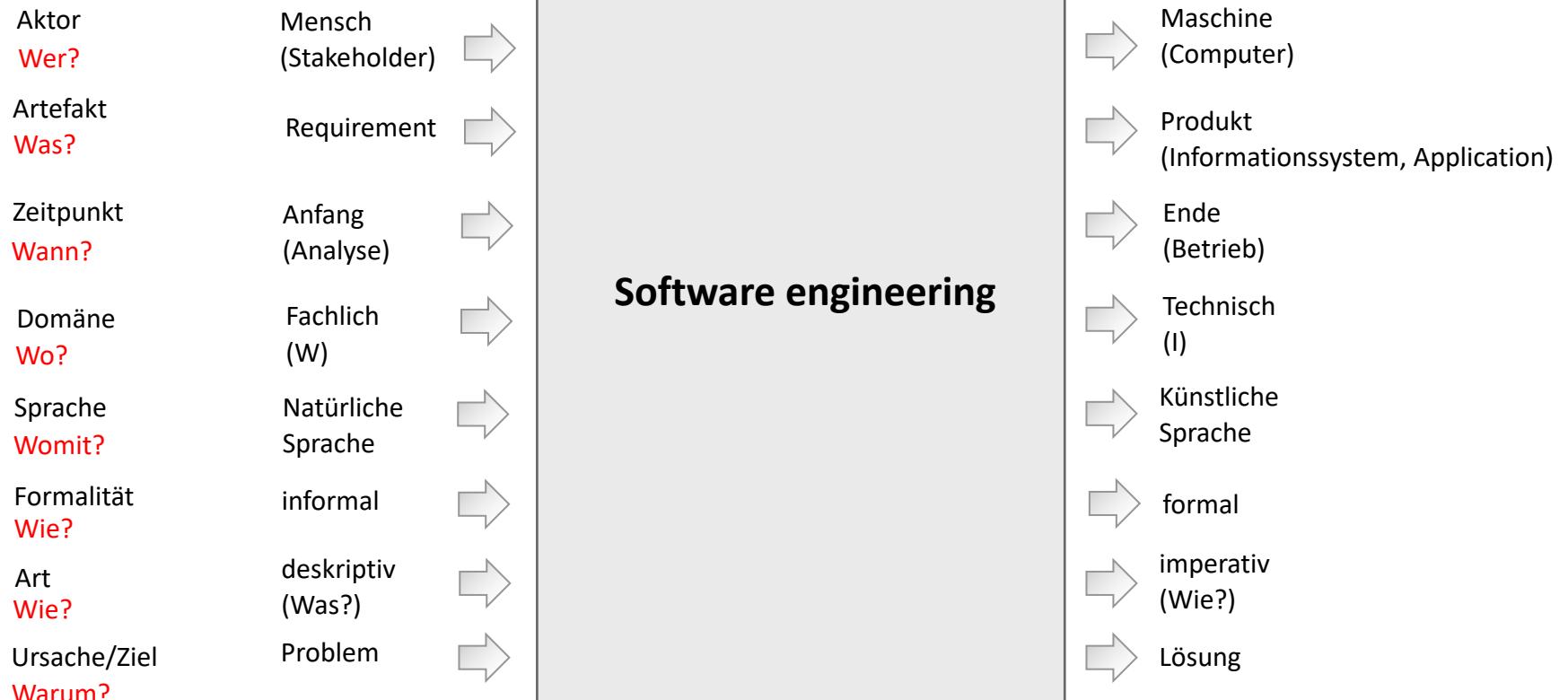
## Legende

Konzept

Beispiel

Beziehung

W-Frage?



# Requirements Engineering

## Fragen am Anfang

- **Was** sind Anforderungen?
- **Warum** gibt es Anforderungen?
- **Wer** stellt Anforderungen?
- **Welche** Anforderungen werden an ein zu entwickelndes Softwaresystem generell gestellt?
- **Wie** sollen Anforderungen definiert werden?
- **Wann** werden Anforderungen thematisiert?
- **Wo** werden Anforderungen festgehalten?

## Fragen am Anfang

- **Was** sind Anforderungen?  
→ Aussagen über Eigenschaften des zukünftigen Systems
- **Warum** gibt es Anforderungen?  
→ Bedürfnisse der Stakeholder
- **Wer** stellt Anforderungen?  
→ Stakeholder
- **Welche** Anforderungen werden an ein zu entwickelndes Softwaresystem generell gestellt?  
→ funktionale / Qualitätsanforderungen (nicht-funktionale Anforderungen)

## Fragen am Anfang

- **Wie** sollen Anforderungen definiert werden?  
→ SMART
- **Wann** werden Anforderungen thematisiert?  
→ funktionale und Qualitätsanforderungen (nicht-funktionale) in allen Phasen
- **Wo** werden Anforderungen festgehalten?  
→ Lasten-/Pflichtenheft (plan-basiert)  
→ Product backlog (agil)

## Wahrnehmung

„You don't perceive objects as they are. You perceive them as you are.“

Eagleman, D.; The Brain - The Story of You; p.35, 2015

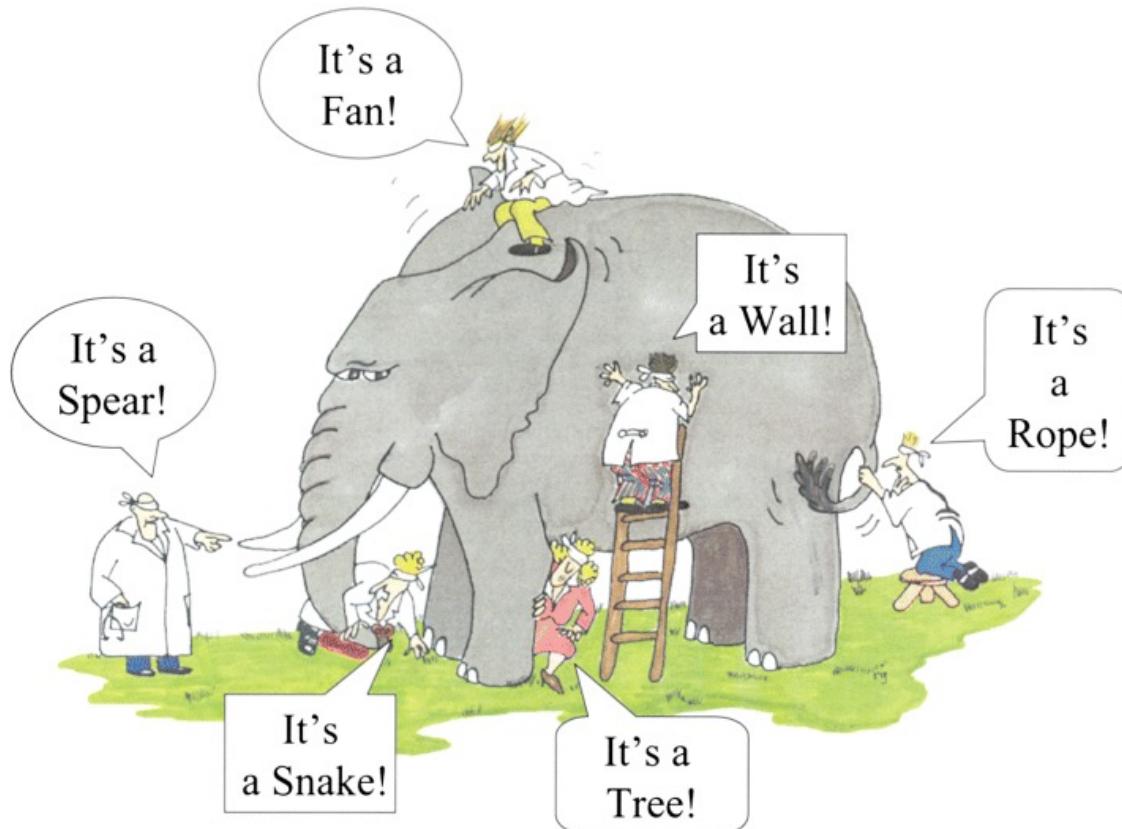
"Das einzige bewusstseinserweiternde Mittel, um mehr zu sehen, ist kritisches Denken, Fragen, Verstehen, Zweifeln ... weil man seine eigene Wahrnehmung auf den Prüfstand stellen muss."

<https://www.brandeins.de/magazine/brand-eins-wirtschaftsmagazin/2019/wahrnehmung/schwindelfrei>; 26.11.2019

## Die Blinden und der Elefant

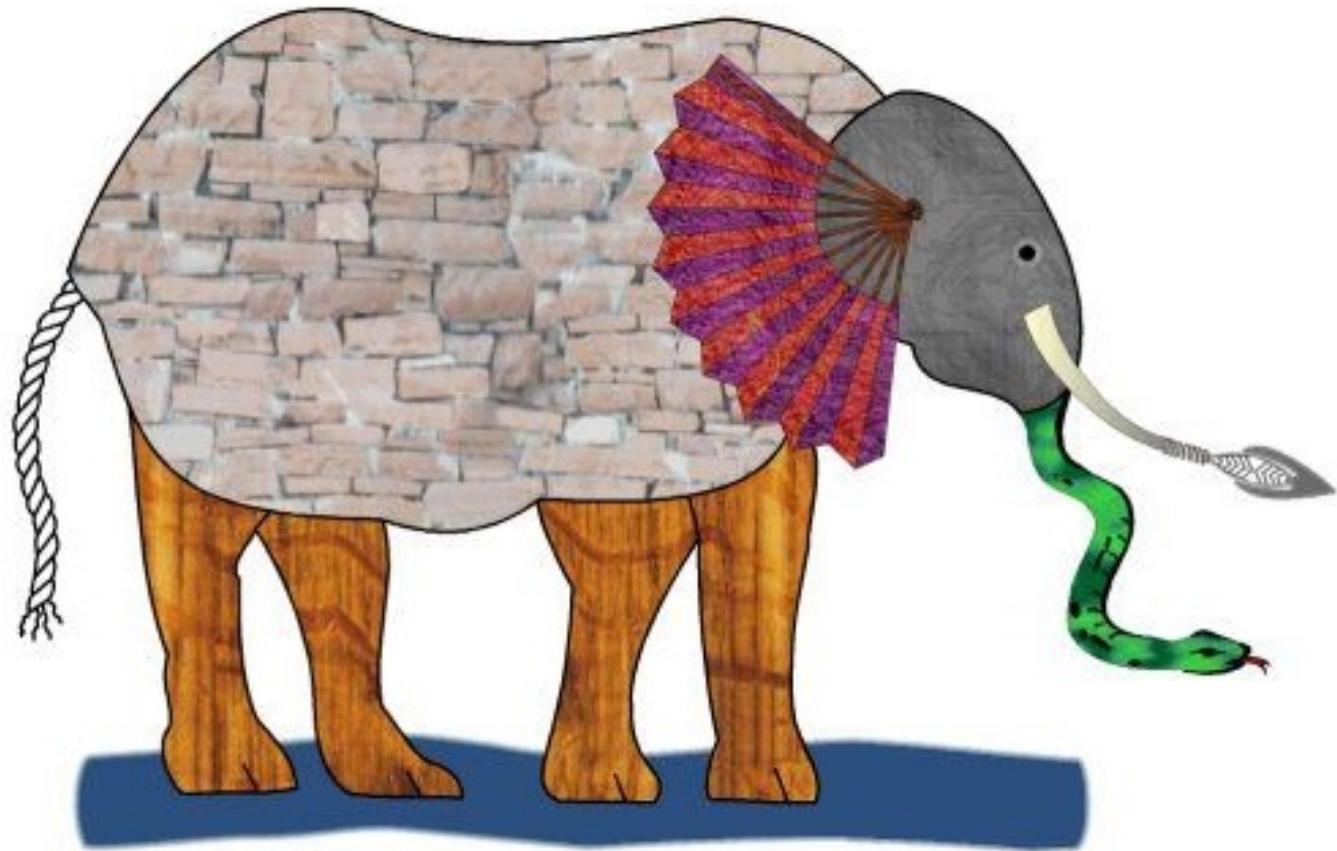
In einem fernen Land stritten sich die Gelehrten einmal darüber, was Wahrheit ist.

Der König, ein wirklich weiser Mann, rief daraufhin einige Blinde zu sich und bat sie, einen Elefanten zu betasten. Danach fragte er, was denn ein Elefant ist.



Quelle: <http://lou.econ.uni-hamburg.de/wordpress/wp-content/wp-uploads//2010/01/The-Blind-Men-and-the-Elephant-Guzlas.gif>; Zugriff: 19.11.2012

## Die Blinden und der Elefant



Quelle: <http://www.organisationsberatung.net/wp-content/uploads//Future-Search-Elefant-7-Blinde.jpg>; Zugriff: 19.11.2012

## Die Blinden und der Elefant

Derjenige, der den Rüssel berührte hatte, sagte, dass ein Elefant wie eine Schlange ist und der Blinde, der die Ohren berührte hatte, sagte, dass ein Elefant ein Fächer sei.

„Nein, das stimmt nicht“, rief ein Anderer, „ein Elefant ist so stämmig wie ein Baum“.

Dieser Blinde hatte die Beine betastet. Der vierte Blinde berichtete, dass seiner Meinung nach ein Elefant lang und glatt und am Ende spitz wie ein Speer ist. Er meinte damit die Stoßzähne.

„Völlig falsch!“, sagte der Nächste, „Ein Elefant ist wie eine Mauer“, denn er hatte die rauhe Haut des Elefanten betastet. Und der Letzte beschrieb den Elefanten als Seil, denn er hatte den Schwanz des Elefanten untersucht.

Schließlich unterbrach der König sie und sagte: „Ihr habt alle recht, aber jeder hat nur ein kleines Stück des Elefanten beschrieben.“

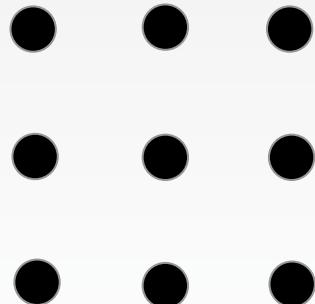
Genauso ist es mit der Wahrheit:

**Was wir sehen oder wahrnehmen, ist oft nur ein kleiner Teil dessen, was wirklich ist**

**Wir sind alle natürlich-blind !!!**

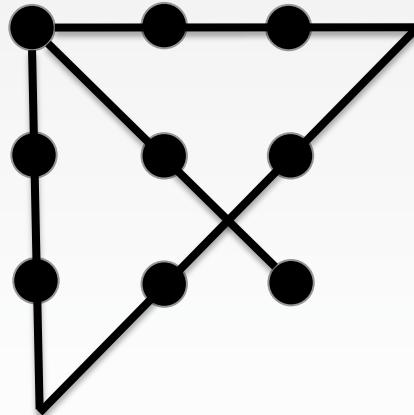
## 9-Punkte-Problem

Alle 9 Punkte sind mit einem Stift durch 4 gerade Linien zu verbinden, ohne den Stift abzusetzen.



## 9-Punkte-Problem

Alle 9 Punkte sind mit einem Stift durch 4 gerade Linien zu verbinden, ohne den Stift abzusetzen.



Russisch?

ЬЛЫГЕ

СЧЕЛ

Russisch

БУРГЕ

СҮЕЛ

PURPLE

GREY

- ➔ Systemanalyse ist Kontextabhängig (vorgeprägt)!!!
- ➔ Manchmal hilft eine andere Sichtweise

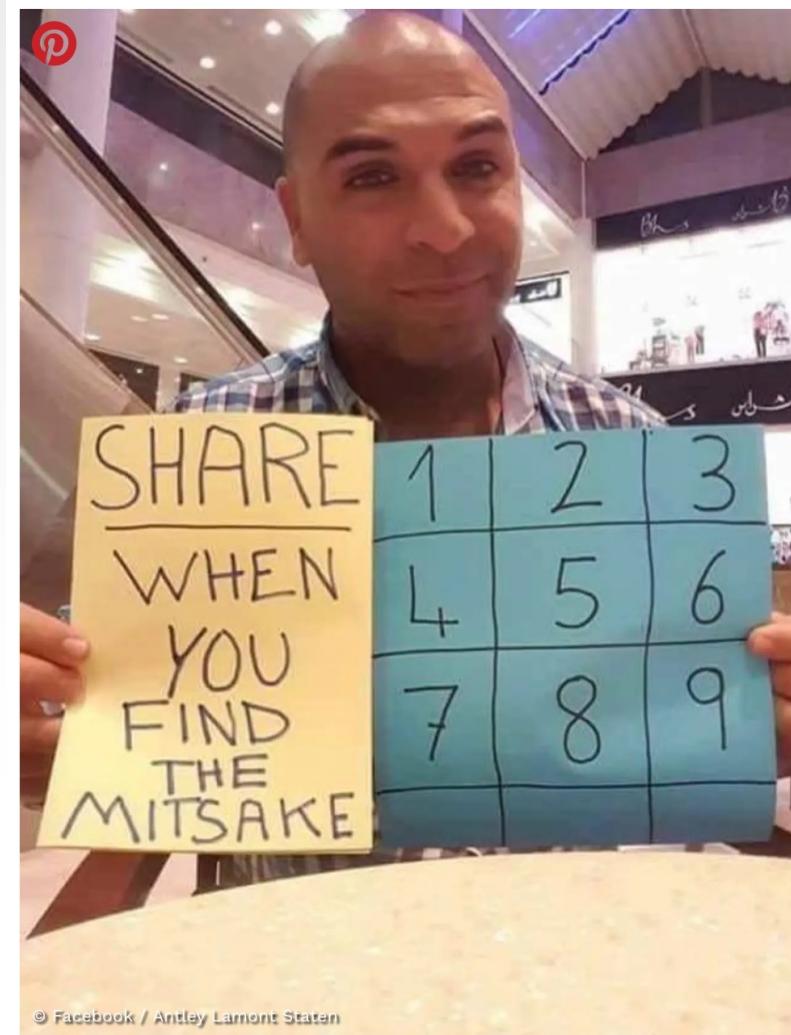
## Licht und Schatten

Beschreiben Sie was Sie sehen!



Quelle: Frings, S., Müller, F.; Biologie der Sinne: Vom Molekül zur Wahrnehmung, Springer Spektrum, 2013

Wo ist der Fehler?



## Annahmen (aus Erfahrung)

Was ist das? Welche Funktion hat das?



Quelle: [http://i00.i.aliimg.com/photo/v0/60097061778\\_1/Toilet\\_paper\\_wholesale.jpg\\_220x220.jpg](http://i00.i.aliimg.com/photo/v0/60097061778_1/Toilet_paper_wholesale.jpg_220x220.jpg);  
[http://i01.i.aliimg.com/img/pb/057/151/268/1285041981629\\_hz-mylibaba-temp12\\_1413.jpg](http://i01.i.aliimg.com/img/pb/057/151/268/1285041981629_hz-mylibaba-temp12_1413.jpg)  
Zugriff: 16.09.2015

## Annahmen (aus Erfahrung)

Was ist die Umgebung? Wie wird es benutzt?



Quelle: [http://www.stylepark.com/db-images/cms/agape/img/p278247\\_488\\_336-1.jpg](http://www.stylepark.com/db-images/cms/agape/img/p278247_488_336-1.jpg)

<https://megabad-static.s3.amazonaws.com/img/00/51/29/hersteller-zack-kuechenbedarf-cuna-kuechenpapierhalter-bildgross2-512948.jpg>

<http://s2.hygi.de/images/artikel/87660921755ae42d575c9c.jpg>

Zugriff: 16.09.2015

04.07.24

© Thomas Slotos

230

## Annahmen (aus Erfahrung)

Was ist die Umgebung? Wie wird es benutzt?



## Annahmen (aus Erfahrung)

Was ist die Umgebung? Wie wird es benutzt?



## Annahmen (aus Erfahrung)

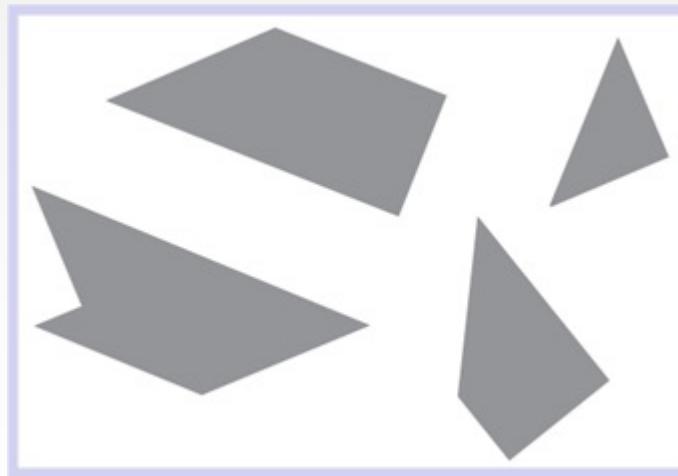
Was ist die falsche und was die richtige Annahme?



Quelle: <https://www.massage-expert.de/bilder/produkte/klein/Handtuchpapier-Rolle-1-lagig-ohne-Kern-20-cm-x-270-m-6-Stueck.jpg>  
Zugriff: 16.09.2015

## Das verflixte T

Setzen Sie die 4 Teile so zusammen, dass ein großes T entsteht!



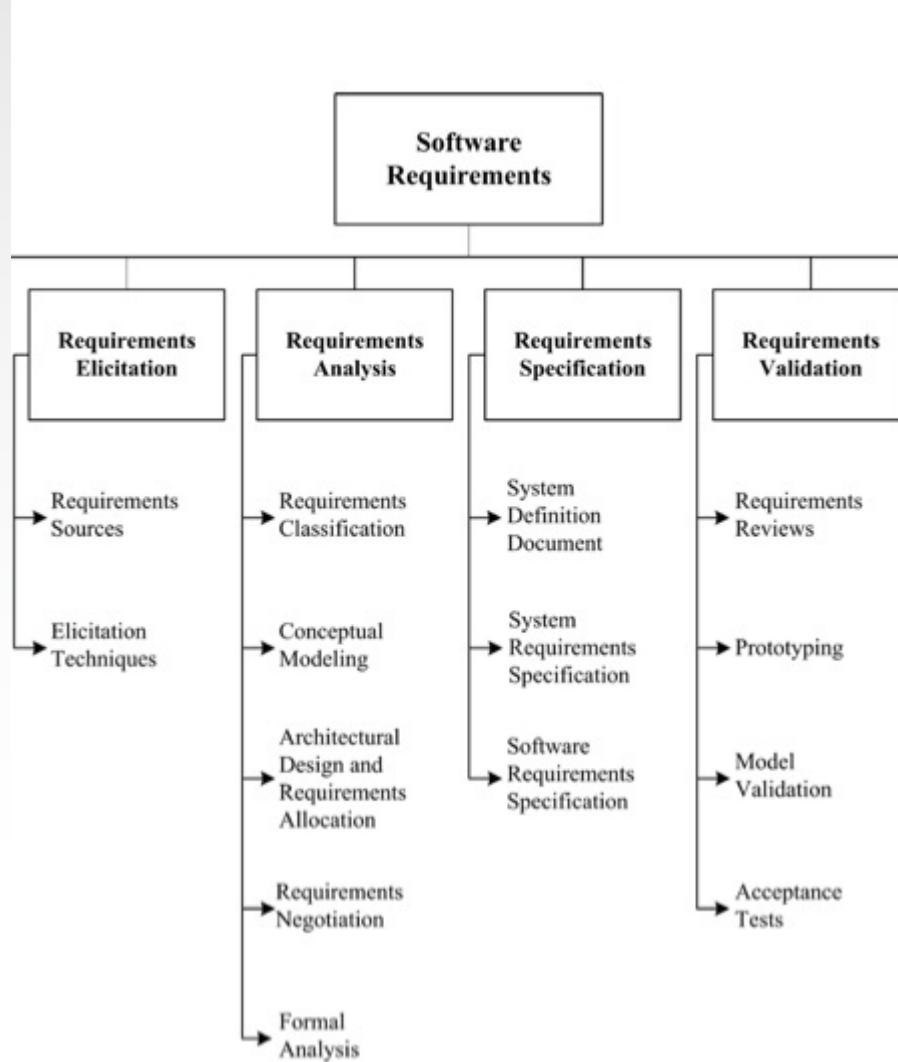
Quelle: <http://www.euroscience.de/tuefteln-und-knobeln/exponate/2d-puzzle/das-verflixte-t>  
Zugriff: 16.09.2015

## Requirements Engineering umfasst:

- Systemanalyse (requirements analysis)  
d.h. eine Anforderungsanalyse in der Anforderungen gesammelt und untersucht werden
- Produktdefinition (requirements specification)  
d.h. eine Anforderungsspezifikation in der Anforderungen gegliedert werden
- Methoden, Beschreibungsmittel und Werkzeuge zur Ermittlung, Analyse und Formulierung von Anforderungen an Softwaresysteme

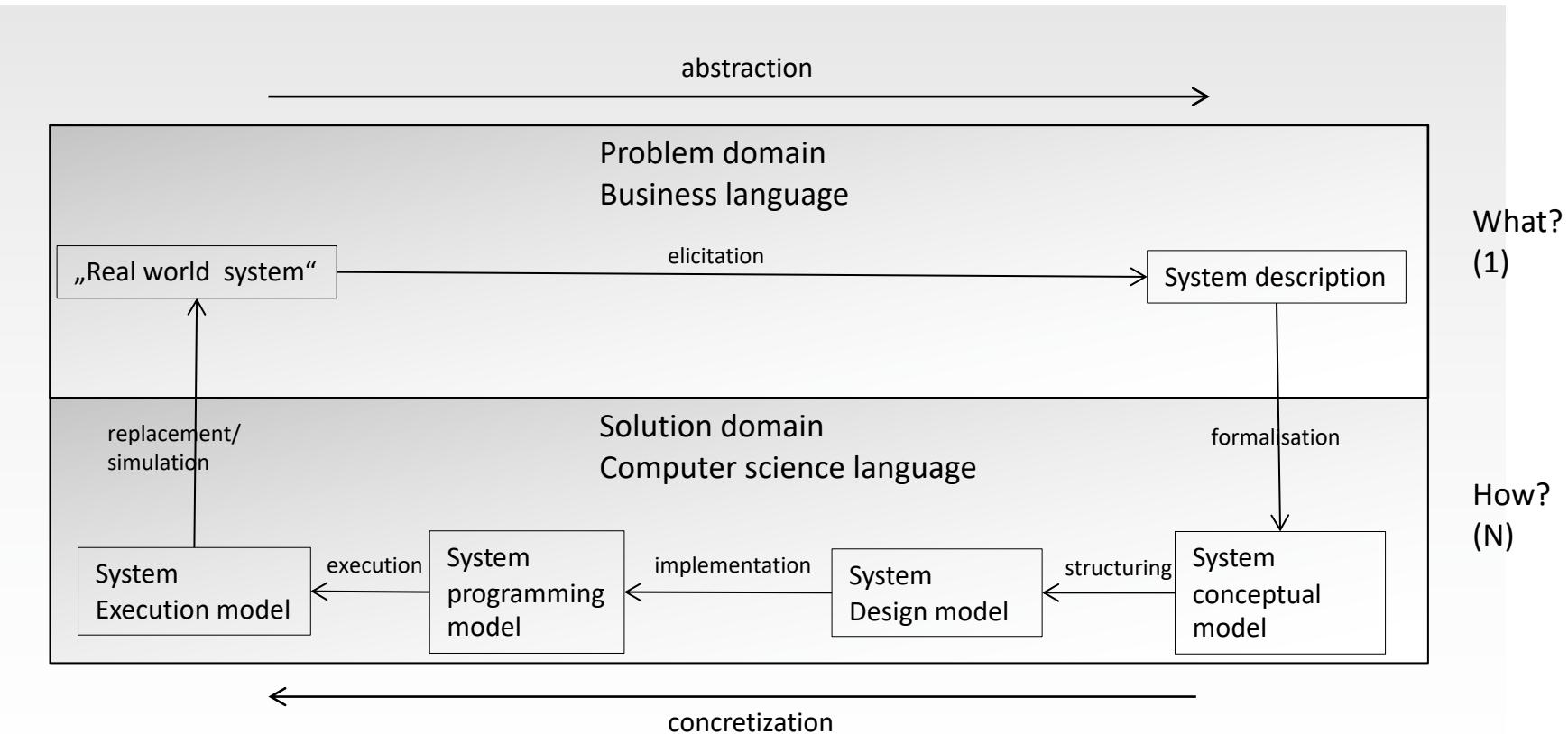
Siehe auch <http://cs.uni-muenster.de/sev/teaching/ws1011/se/SE1011-Kap4.pdf>

# Requirements Engineering – Definition (2)



Quelle: Bourque, P., Fairley, R.; SWEBOK V3.0 – Guide to the Software Engineering Body of Knowledge, IEEE, 2004  
<https://www.computer.org/education/bodies-of-knowledge/software-engineering>; Zugriff: 30.11.2020

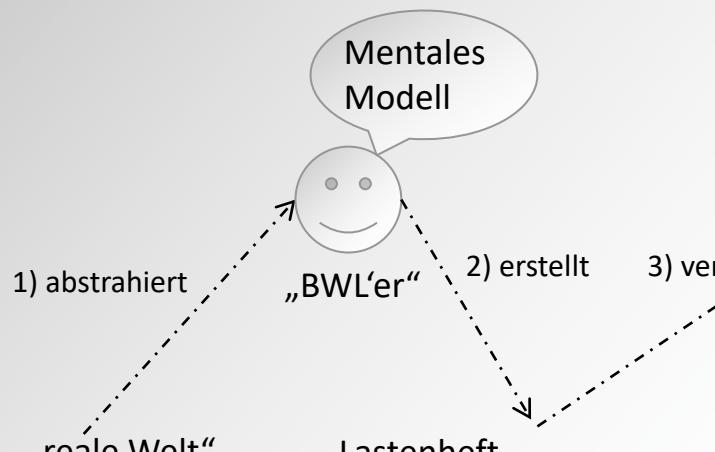
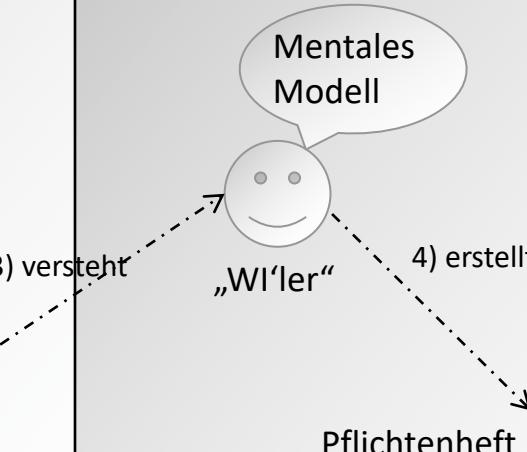
# Requirements engineering - Problem domain – Solution domain (1)



„First, the concepts which are formed in the mind function as a system of mental representation which classifies and organizes the world into meaningful categories. If we have a concept for something, we can say we know its ‚meaning‘. But we cannot communicate this meaning without a second system of representation, a language.“

Quelle: Hall, R.; Representation: Cultural Representation and Signifying Practices, 2003

## Plan-basiertes Vorgehensmodell - Lasten-/Pflichtenheft

Problem domain Business language	Solution domain Computer science language
 <p>1) abstrahiert        „reale Welt“</p> <p>„BWL'er“</p> <p>Mentales Modell</p> <p>2) erstellt        Lastenheft</p> <ul style="list-style-type: none"> <li>• Fachsprache Wirtschaft</li> <li>• grobe Anforderungsbeschreibung</li> <li>• informal</li> <li>• Machbarkeitsgrundlage</li> </ul>	 <p>3) versteht        „WI'ler“</p> <p>Mentales Modell</p> <p>4) erstellt        Pflichtenheft</p> <ul style="list-style-type: none"> <li>• Fachsprache Informatik</li> <li>• verfeinerte Anforderungsbeschreibung</li> <li>• semi-formal</li> <li>• Vertragsgrundlage</li> </ul>
What? (1)	How? (N)

„Analysis is the study and modeling of a given problem domain, within the context of stated goals and objectives.

It focuses on *what* a system is supposed to do, rather than *how* it is supposed to do it [...]. The components of the problem domain can be described as anything that the end users of the system, both humans and machines, view as part of the problem context.”

Quelle: Rubin, K. S., & Goldberg, A. (1992). Object behavior analysis. *Communications of the ACM*, 35(9), 48-62.

“During analysis the focus is on what needs to be done, independent of how it is done.”

Quelle: Rumbaugh, J. et al.; Object-Oriented Modeling and Design, 1991

“The system requirements model is a technology-independent model of the problem the system is to solve: it represents the *what*. The system architecture model is a technology-dependent model of the solution to the problem. It represents the *how*.”

Quelle: Hatley, D.; et al.; Process for system architecture and requirements engineering, 2000

“A problem is a difference between what is perceived to be the case and what is desired, that we want to reduce; a solution is an action that reduces this difference.”

Quelle: Wieringa, R. J. (2004). Requirements engineering: Problem analysis and solution specification. In *Web Engineering* (pp. 13-16). Springer Berlin Heidelberg.

“Analysis means the process of extracting the "needs" of a system - what the system must do to satisfy the client, not how it should be done.”

Quelle: Coad, P., Yourdon, E.; Object-oriented Analysis, 1990

„This combination of descriptive and imperative languages provides separate descriptions of *what* is to be done (the specification) and *how* it is to be done (the implementation).“

Quelle: Shaw, M.; Abstraction: Imposing order on complexity in software design, 2004

## System analysis phase

“The analysis phase aims to analyze, specify, and define the system to be built. The models developed in this process will describe what the system is to do. Two models are developed: the requirements model and the analysis model. Both these models are logical in the sense that they do not incorporate any implementation details. This will guarantee that the system's architecture will be based on the problem domain and not on conditions prevailing during implementation.

...

The requirements model is produced in the analysis phase, and aims to delimit the system and define the functionality it should offer. This model could function as a contract between the developer and the orderer of the system and thus formalizes the developer's view of what the customer wants. The requirements model consists of:

- A use case model
- ...
- A problem domain model

...

The analysis model aims to structure the system independently of the actual implementation environment. This means focus is on the logical structure of the system.”

Quelle: Jacobson , I.; Object-Oriented Software Engineering - A Use Case Driven Approach, 1993

„Bei der Systemanalyse konstruiert der Betrachter des Systems ein Modell aus Anwendersicht. Dabei trifft er eine Auswahl bezüglich der relevanten Elemente und Beziehungen des Systems.“

Dieses Modell ist ein begrenztes, reduziertes, abstrahiertes Abbild der Wirklichkeit, mit dessen Hilfe Aussagen über vergangene und zukünftige Entwicklungen und Verhaltensweisen des Systems in bestimmten Szenarien gemacht werden sollen.“

Quelle: Knopper, K. Software Engineering;  
<http://www.knopper.net/bw/se/vorlesung/vorlesung-print.pdf>; Zugriff: 03.11.2014

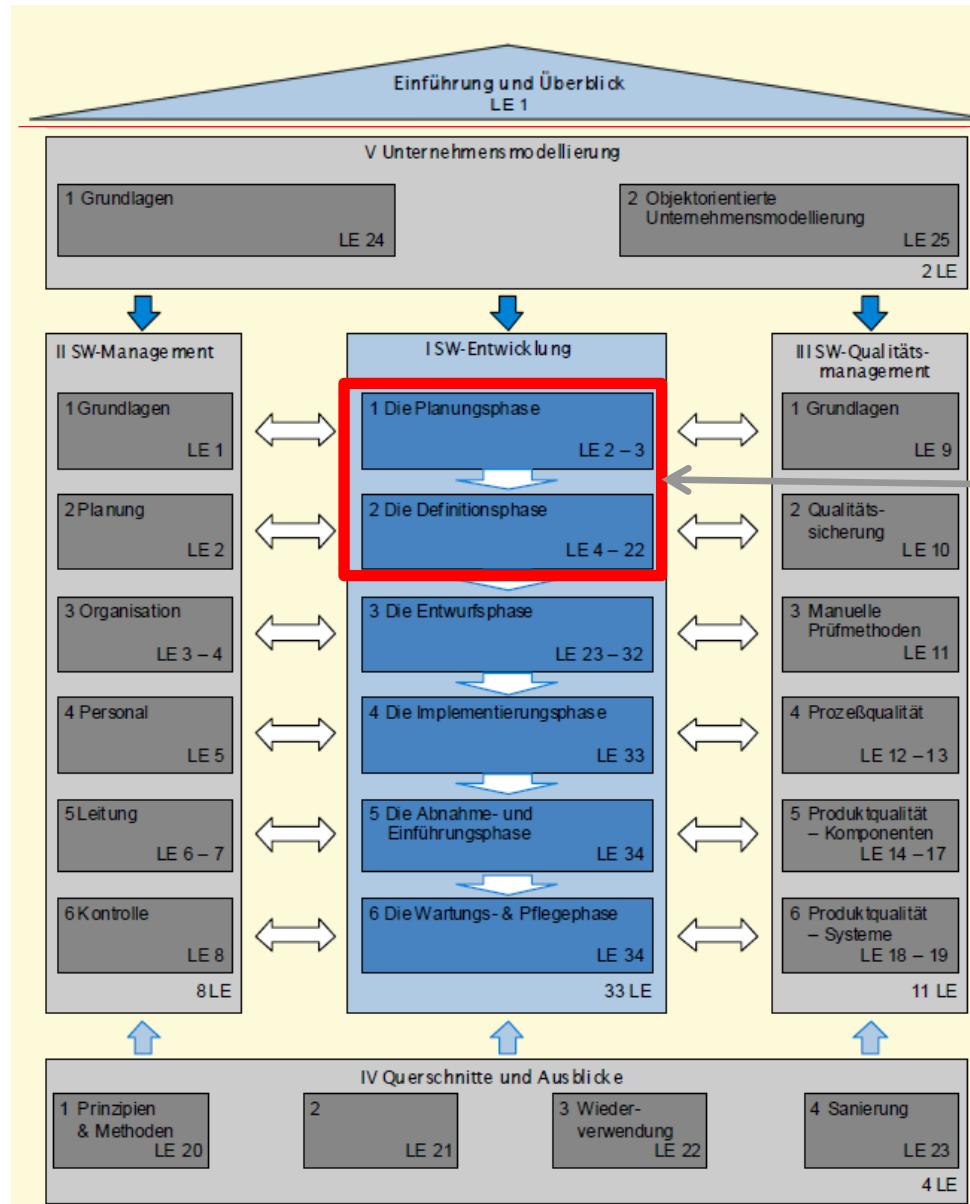
## Conceptual modeling

„... conceptual modelling is the process of abstracting a model from the real world. The modeller is presented with a problem situation that is amenable to simulation modelling and then has to determine what aspects of the real world to include, and exclude, from the model, and at what level of detail to model each aspect. These decisions should generally be a joint agreement between the modeller and the problem owners i.e. the stakeholders who require the model to aid decision-making.“

Quelle: Kotiadis, K.; Robinson, S.; Conceptual modelling: Knowledge Acquisition and Model Abstraction; Proc. Of the 2008 Winter Simulation Conference;  
<http://www.informs-sim.org/wsc08papers/114.pdf>; Zugriff: 10.11.2014

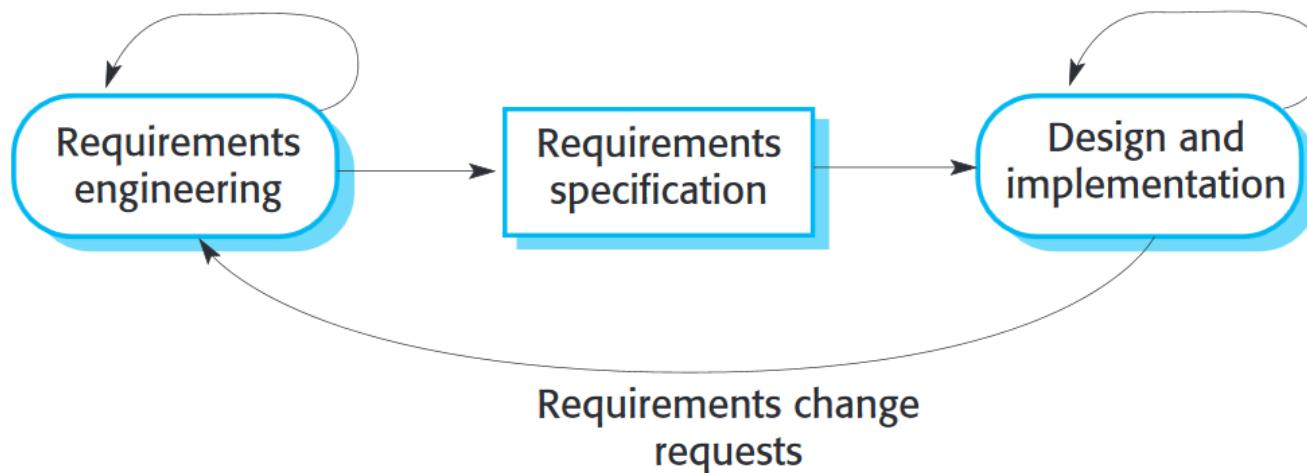
# Requirements engineering - Planungs- und Definitionsphase (1)

## Verortung in der Softwaretechnik

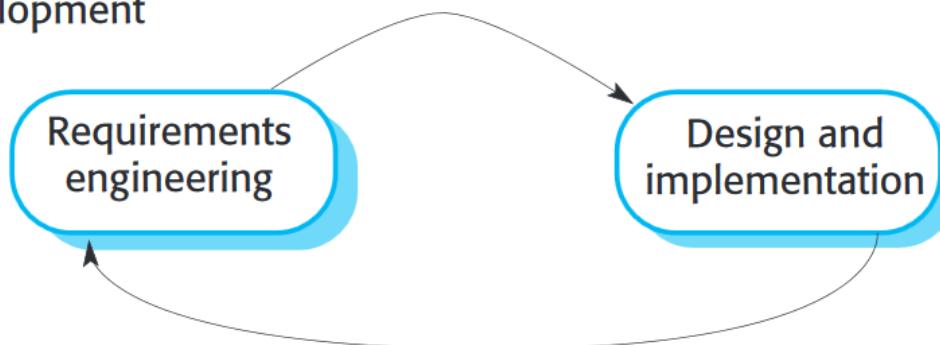


## Vorgehensweisen

### Plan-based development

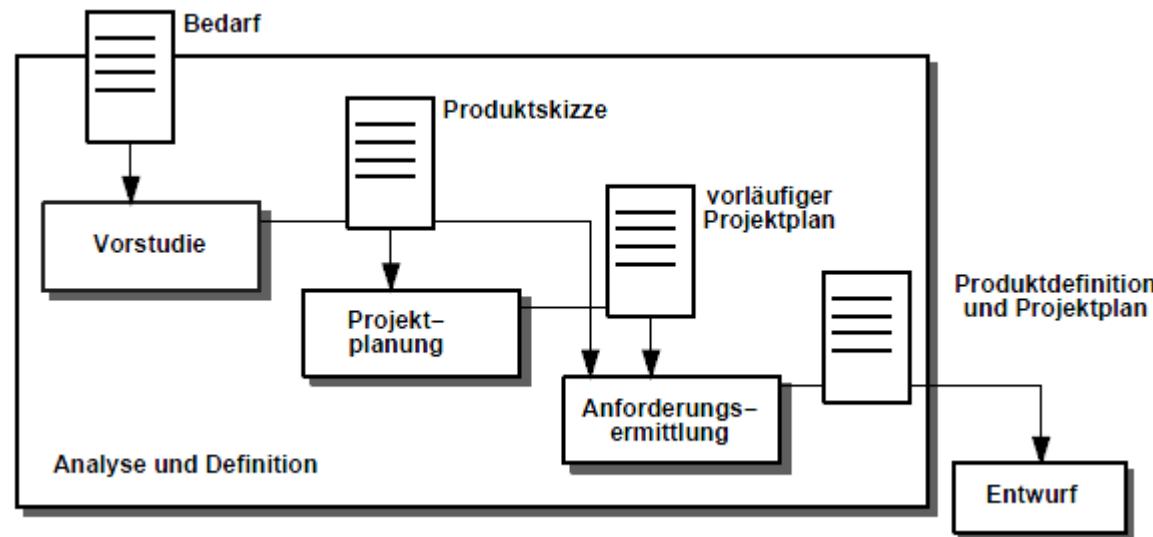


### Agile development



## Vorgehensweisen

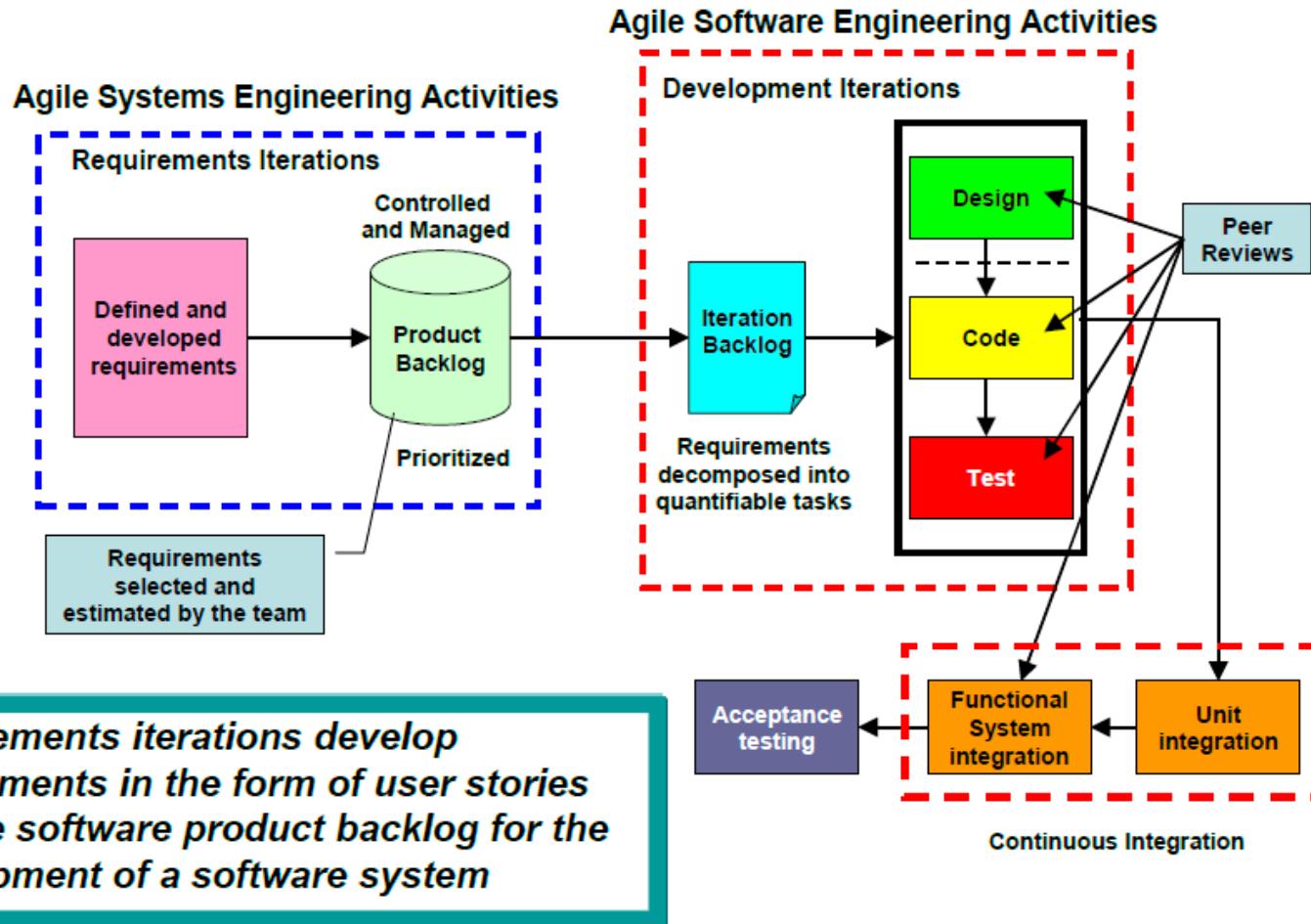
### Plan-based development



Quelle: unbekannt

## Vorgehensweisen

### Agile development



## Vergleich Planbasiertes vs. Agiles Vorgehen

	Planbasiert	Agil
<b>Prozess</b>	vorhersagbar	adaptiert
<b>Ausrichtung</b>	An Prozessvorgabe des Produktionsablauf	An situativen Wünschen des Kunden
<b>Kommunikation</b>	formal	informell
<b>Dokumentation</b>	umfangreich, formal	minimal, semiformal
<b>Timeboxing</b>	nein	häufig
<b>Kundenbeteiligung</b>	Nur am Anfang und Ende	Durchgehend
<b>Kosten/Preis</b>	Festpreis	Nach Aufwand

## Vergleich Planbasiertes vs. Agiles Vorgehen

Factor	Agility discriminators	Plan-driven discriminators
Size	Well matched to small products and teams; reliance on tacit knowledge limits scalability.	Methods evolved to handle large products and teams; hard to tailor down to small projects.
Criticality	Untested on safety-critical products; potential difficulties with simple design and lack of documentation.	Methods evolved to handle highly critical products; hard to tailor down efficiently to low-criticality products.
Dynamism	Simple design and continuous refactoring are excellent for highly dynamic environments, but present a source of potentially expensive rework for highly stable environments.	Detailed plans and "big design up front" excellent for highly stable environment, but a source of expensive rework for highly dynamic environments.
Personnel	Require continuous presence of a critical mass of scarce Cockburn Level 2 or 3 experts; risky to use nonagile Level 1B people.	Need a critical mass of scarce Cockburn Level 2 and 3 experts during project definition, but can work with fewer later in the project—unless the environment is highly dynamic. Can usually accommodate some Level 1B people.
Culture	Thrive in a culture where people feel comfortable and empowered by having many degrees of freedom; thrive on chaos.	Thrive in a culture where people feel comfortable and empowered by having their roles defined by clear policies and procedures; thrive on order.

Boehm, B., Turner, R., Using Risk to Balance Agile and Plan-Driven Methods, 2003

## Vergleich Planbasiertes vs. Agiles Vorgehen

	Software Process Model	Advantages	Disadvantages
Plan Driven	<ul style="list-style-type: none"> <li>• Waterfall</li> <li>• Incremental Development</li> <li>• Iterative development</li> <li>• Spiral Development</li> <li>• Prototype Model</li> <li>• Rapid Application Development</li> </ul>	<ul style="list-style-type: none"> <li>• Suitable for large systems and teams.</li> <li>• Handles highly critical systems effectively.</li> <li>• Appropriate for stable development environment.</li> <li>• Require experienced personnel at the beginning.</li> <li>• Success achieved through structure and order.</li> </ul>	<ul style="list-style-type: none"> <li>• Longer length in each iteration or increment.</li> <li>• Cannot accommodate changes any time.</li> <li>• Lack of user involvement throughout the life cycle of the product.</li> <li>• Costly for the dynamic development environment.</li> <li>• Assume that, future changes will not occur.</li> </ul>
Agile	<ul style="list-style-type: none"> <li>• Scrum model</li> <li>• Extreme Programming (XP)</li> <li>• Dynamic System Development Method</li> <li>• Kanban</li> <li>• Feature Driven Development</li> </ul>	<ul style="list-style-type: none"> <li>• Suitable for small to medium systems and teams.</li> <li>• Can accommodate changes at any time.</li> <li>• Effective for the dynamic development environment.</li> <li>• Required expert agile personnel throughout the life cycle.</li> <li>• Success achieved through freedom and chaos.</li> </ul>	<ul style="list-style-type: none"> <li>• Not suitable for large systems (except FDD).</li> <li>• Shorter length in each iteration.</li> <li>• Can accommodate changes at any time.</li> <li>• Costly for the stable development environment.</li> <li>• Assume that, frequent future changes will occur.</li> </ul>

## Schritte

- „1. Festlegen der Systemgrenzen zur Unterscheidung von System und Umwelt.
- 2. Feststellen derjenigen Systemelemente, die für die Fragestellung als relevant betrachtet werden.
- 3. Feststellen derjenigen Beziehungen zwischen den Systemelementen, die für die Fragestellung als relevant betrachtet werden.
- 4. Feststellen der Systemeigenschaften auf der Makroebene.“

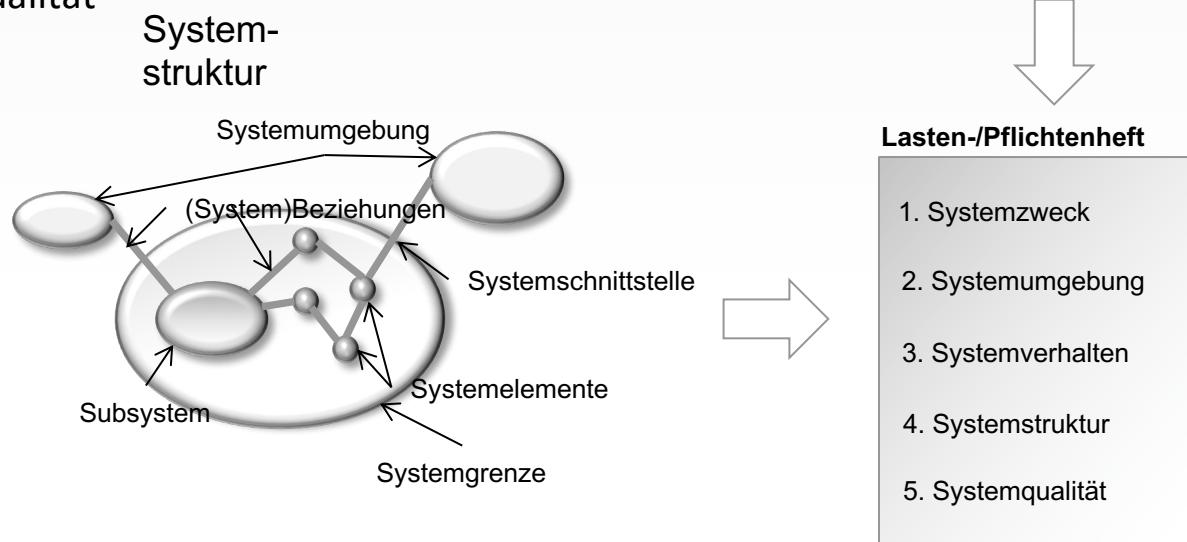
Quelle: Knopper, K. Software Engineering;  
<http://www.knopper.net/bw/se/vorlesung/vorlesung-print.pdf>; Zugriff: 03.11.2014

## Plan-basiertes Vorgehensmodell - Dokumente

Lasten- und Pflichtenheft (engl. Software Requirements Specification) sind Dokumente zur Systemspezifikation. D.h. in beiden Dokumenten wird das zukünftige Informationssystem beschrieben.

Beide Dokumente enthalten Beschreibungen zu folgenden Punkten:

- Zweck
- Umwelt (andere Systeme)
- Verhalten
- Struktur
- Qualität



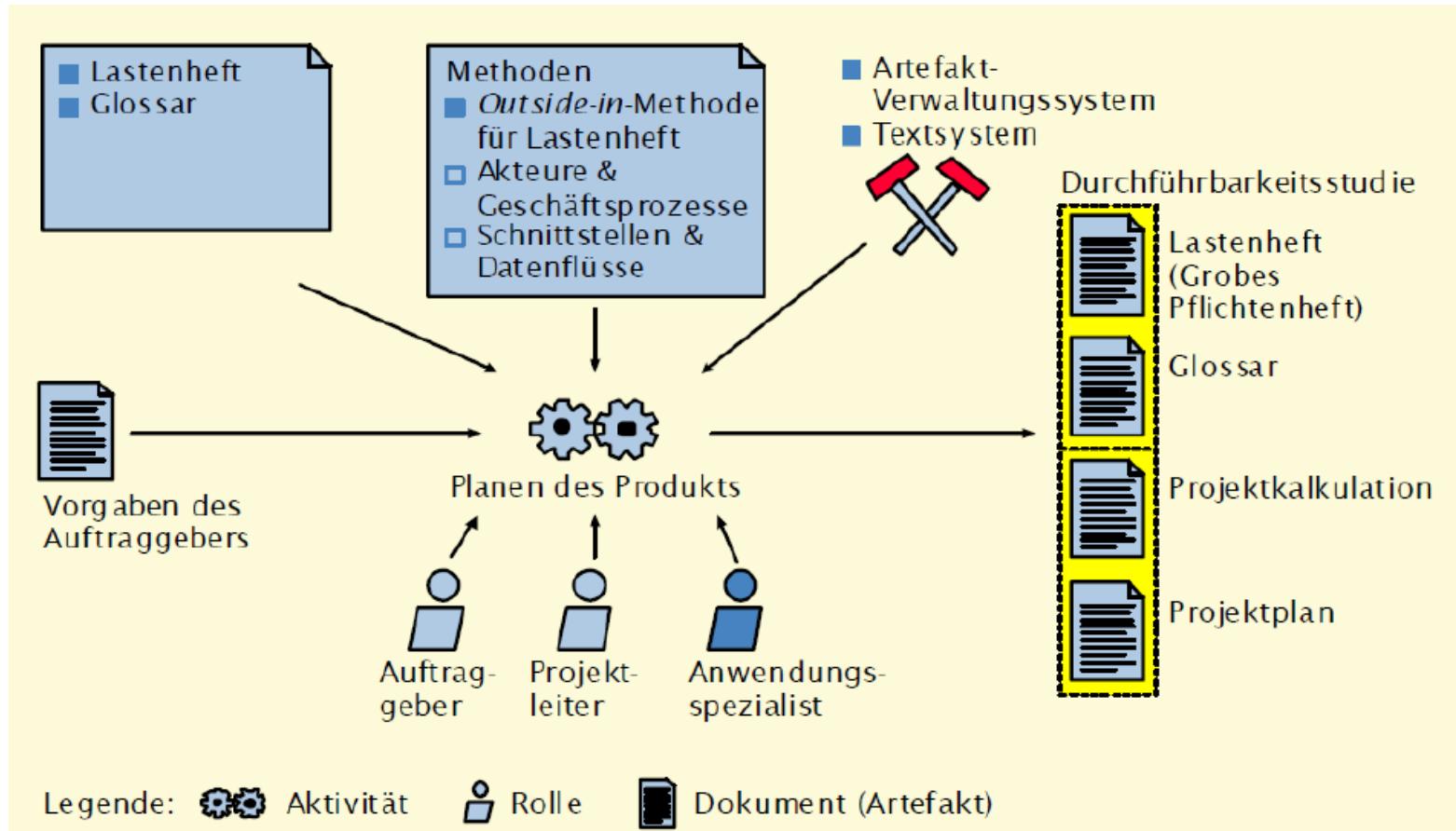
## Systemanalyse (requirements engineering)

- Systematische Vorgehensweise, um die Anforderungen in einem iterativen Prozess zu ermitteln

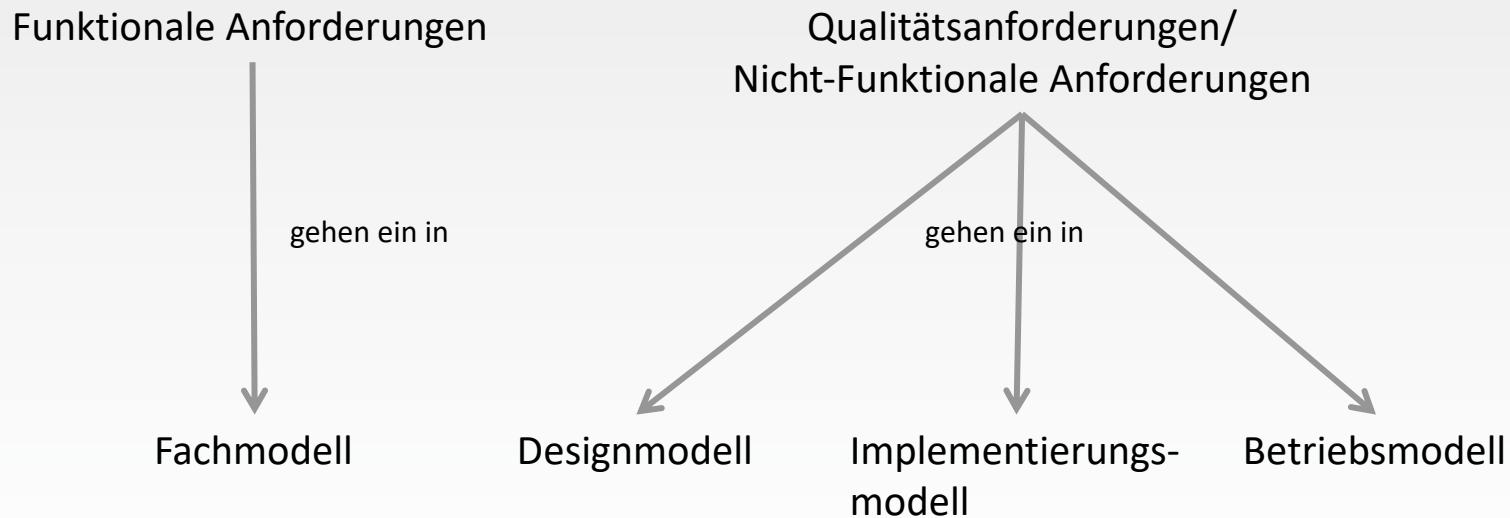
## Anforderungen (requirements)

- Legen die qualitativen und quantitativen Eigenschaften eines Produktes aus Sicht des Auftraggebers fest

## Planungsphase - Übersicht



Quelle: Balzert, H.; Lehrbuch der Softwaretechnik, 2001



## Beispiele

Funktionale Anforderungen

Fachmodell

„Die Anwendung beinhaltet die Funktionen A, B, C.“

Qualitätsanforderungen/Nicht-Funktionale Anforderungen

Designmodell

„Die Anwendung muss so flexibel sein, dass sie um weitere Funktionen im Bereich XY erweiterbar ist.“

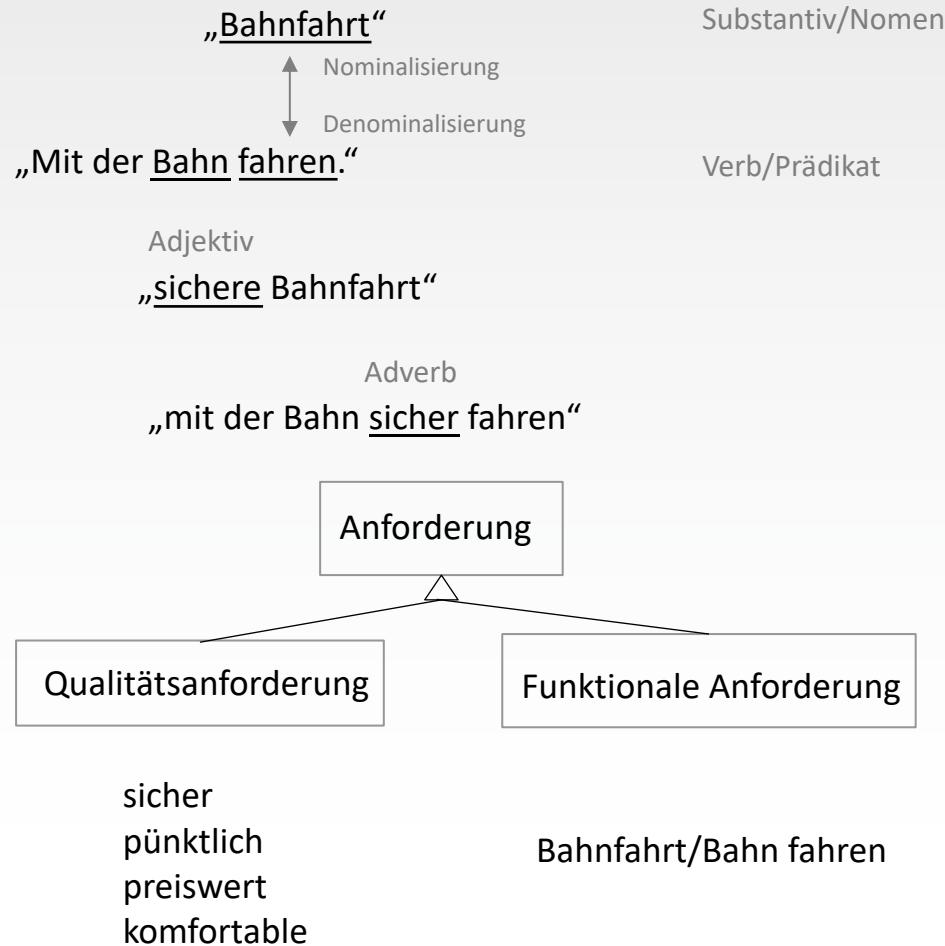
Implementierungsmodell

„Die Anwendung soll in einer objekt-orientierten Programmiersprache entwickelt werden um die Fachkonzepte möglichst identisch abzubilden.

Betriebsmodell

„Die Anwendung soll in allen gängigen Web-Browsern laufen und mit dem Web-Server ABC kommunizieren.“

## Natürliche Sprache



See also: Hussain, I.; et al.; Using Linguistic Knowledge to Classify Non-functional Requirements in SRS documents, 2008

## Anforderungen und Softwareproduktqualität

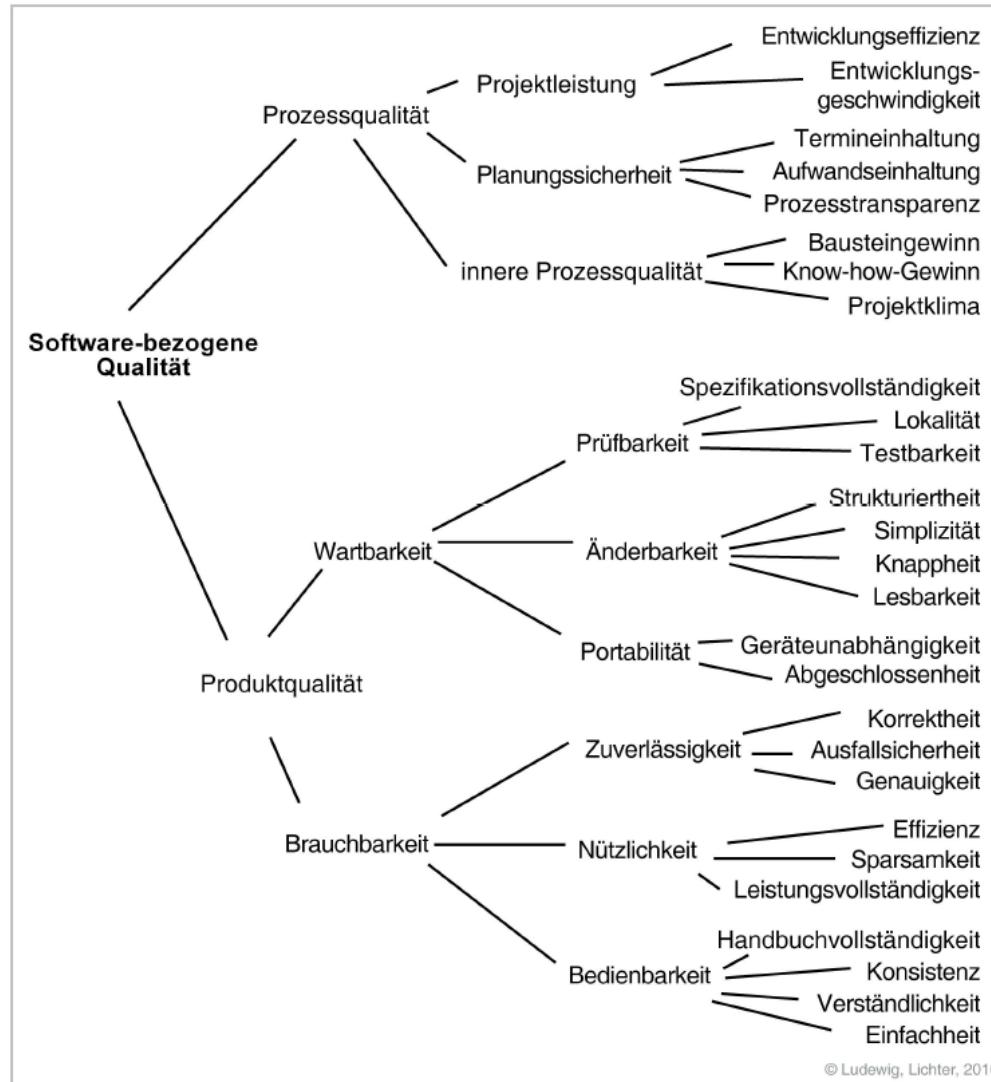


Quelle: ISO / IEC 25010; <http://iso25000.com/index.php/en/iso-25000-standards/iso-25010>; Zugriff: 11.12.2017

„Quality of Service (QoS) is a general term that covers system ... [operation quality], as opposed to system operation. A system will be built to ... [execute] some set of functions for its users. These functions can be called the operational or functional features of the system; ... But the ... [execution] of each function will take time, require system resources, and be subject to occasional system errors or failures: these and other similar features are ... non-functional features of the system, which come under the heading of QoS. In general, the users of a system will have requirements both for the functions that are to be ... [executed] and for the QoS with which they are ...[executed] ...“

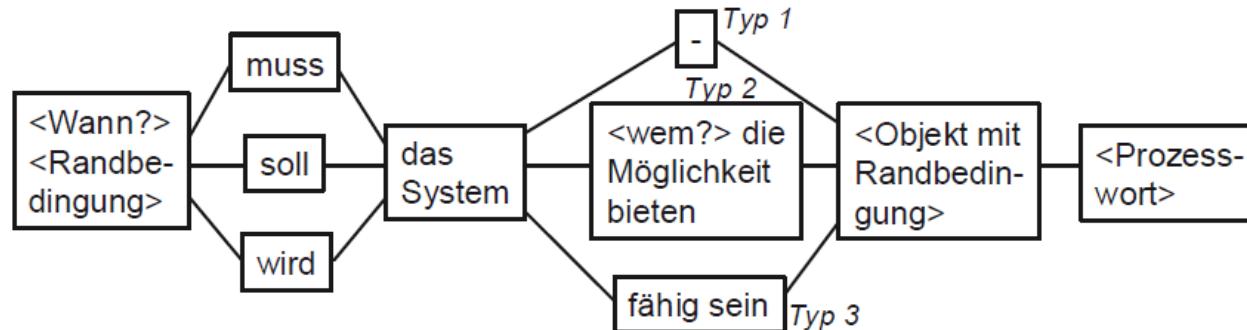
C. Sluman, J. Tucker, J. LeBlanc, B. Wood, "Quality of Service (QoS) OMG Green Paper", Object Management Group, Object and Reference Model Subcommittee of the Architecture Board, OMG Document Number ormsc/97-06-04, December, 1997.

# Requirements engineering - Planungsphase – Anforderungen (5)



Quelle: Ludewig, J., Lichter, H.; Software Engineering, 2010

## Anforderungsdefinition



### Beispiel Typ 1

Das System muss die *Kundendaten permanent speichern*.

### Beispiel Typ 2

Das System muss dem *Kunden* die Möglichkeit bieten, sich über *Seminare und Veranstaltungen* zu informieren.

### Beispiel Typ 3

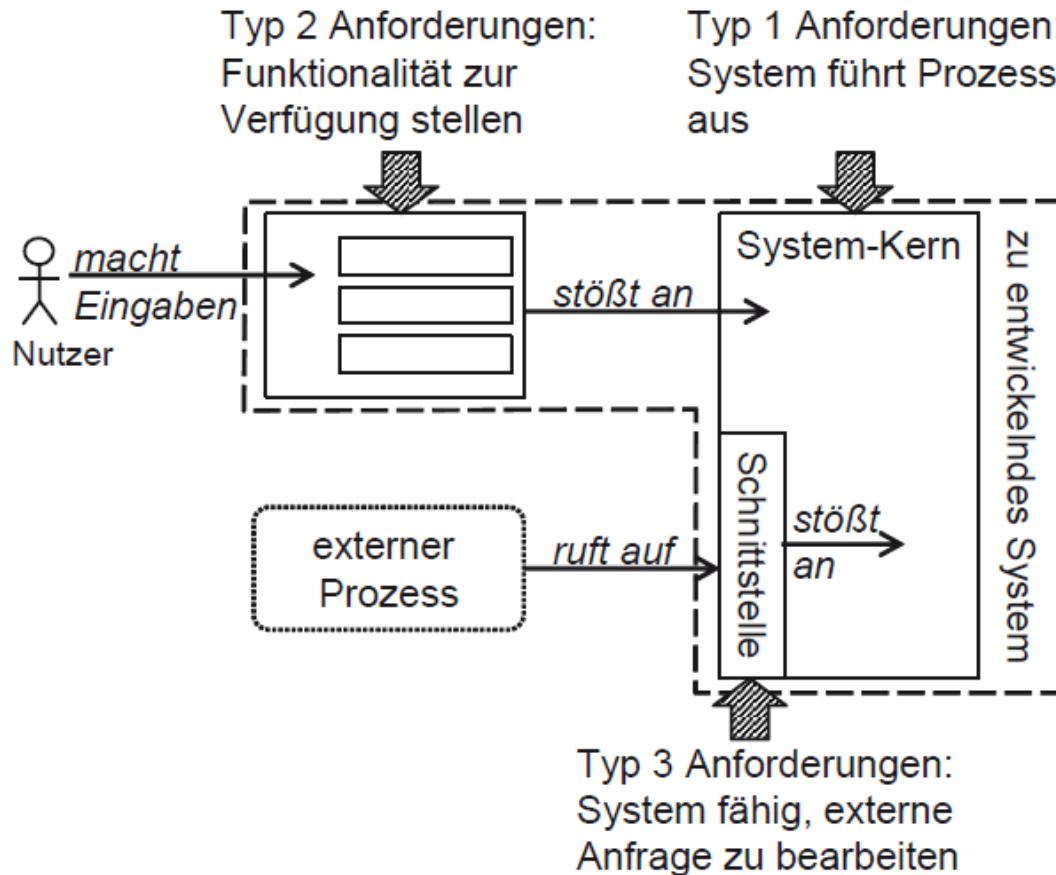
Das System muss fähig sein, dem Buchhaltungssystem *Rechnungsdatensätze mindestens einmal am Tag* zur Verfügung zu stellen

Quelle: Rupp, C., Requirements-Engineering und –Management, 2014

bzw. Balzert, H.; Softwaretechnik – Basiskonzepte und Requirements Engineering, 2009

© Thomas Słotos

## Anforderungsgliederung



## Anforderungsbeschreibung – SMART zur Zieldefinition

Bedeutung	Beschreibung
Spezifisch	Eindeutig /präzise
Messbar	Überprüfbare Kriterien
Akzeptiert	gewollt
Realistisch	fachl. / techn. möglich
Terminierbar	Klare Terminvorgabe

„A requirement is a feature that the system must have or a constraint that it must satisfy to be accepted by the client.“

Quelle: Bruegge, B.; Dutoit A.; Object-Oriented Software Engineering, 3rd. ed. Prentice Hall, 2010

„The requirements for a system are the descriptions of what the system should do— the services that it provides and the constraints on its operation. These requirements reflect the needs of customers for a system that serves a certain purpose ...“

Quelle: Sommerville, I.; Software Engineering, 9th ed. 2011

„[A requirement is a] statement of a customer need or objective, or of a condition or capability that a product must possess to satisfy such a need or objective. A property that a product must have to provide value to a stakeholder.“

Quelle: Wieggers, K., Beatty, J., Software Requirements, 3rd ed. 2013

## Anforderungsarten

„Requirements specify a set of features that the system must have. A **functional requirement** is a specification of a function that the system must support, whereas a **non-functional requirement** is a constraint on the operation of the system that is not related directly to a function of the system.“

Quelle: Bruegge, 2010

„Eine **funktionale Anforderung** legt eine vom Softwaresystem oder einer seiner Komponenten bereitzustellende Funktion oder bereitzustellenden Service fest.“

„**Nichtfunktionale Anforderungen** (non-functional requirements, kurz NFRs), auch Technische Anforderungen oder *Quality of Services* (QoS) genannt, beschreiben Aspekte, die typischerweise mehrere oder alle funktionalen Anforderungen betreffen bzw. überschneiden (*cross-cut*). ... Nicht-funktionale Anforderungen haben einen großen Einfluss auf die Softwarearchitektur.“

Quelle: Balzert, H.; Lehrbuch der Softwaretechnik, 2001

## Anforderungsarten

„... distinguish ...‘user requirements’ to mean the high-level abstract requirements and ‘system requirements’ to mean the detailed description of what the system should do.“

„Domain requirements are derived from the application domain of the system rather than from the specific needs of system users.“

Quelle: Sommerville, I.; Software Engineering, 9th ed.. 2011

## Rollenbegriff in der Softwareentwicklung

Während der Softwareentwicklung werden unterschiedliche Rollen für unterschiedliche Tätigkeiten besetzt. Aufgrund dieser Arbeitsteilung sind jeder Rolle bestimmte Eigenschaften und Aufgaben zugewiesen, was zum einen den Umfang des zu bearbeitenden Bereichs einschränkt, gleichzeitig aber dafür sorgt, dass nur ein Teilausschnitt des Gesamtsystems überblickt wird. Jede Rolle betrachtet das Gesamtsystem aus seiner Rollensicht (**viewpoint**).

Die Rollen, die von Personen oder Organisationseinheiten eingenommen werden, die nicht an der Softwareentwicklung beteiligt sind, werden **Stakeholder** genannt.

Auch innerhalb eines erstellten konzeptuellen Modells wie z.B. im Klassendiagramm der UML sind das Herausarbeiten von Rollen eine wesentliche Aufgabe um Software zu entwickeln, die flexibel angepasst werden kann. Hierzu dienen u.a. sogenannte **Muster**. Muster stellen eine Anordnung von Rollen dar. Für gleichartige Problemstellungen bieten diese Muster aufgrund von verallgemeinerten Rollen eine Lösung für das betrachtete Problem an.

## Interessenvertreter / Stakeholder

„The term stakeholder generalizes the traditional notion of a customer or user in requirements engineering to all parties involved in a system's requirements.“

Quelle: Bruegge, 2010

„A **Stakeholder**‘ is an individual person or other legal entity able to act like a person (e.g. a Limited Company, an Industry Regulator, a Registered Charity) playing one or more Roles.“

Quelle: Alexander, I., A Taxonomy of Stakeholders – Human Roles in System Development, Int. Journal of Technology and Human Interaction, Vol. 1, 2005  
[http://www.scenarioplus.org.uk/papers/stakeholder\\_taxonomy/stakeholder\\_taxonomy.htm](http://www.scenarioplus.org.uk/papers/stakeholder_taxonomy/stakeholder_taxonomy.htm); Zugriff: 11.11.2013

„A *stakeholder* is a person, group, or organization that is actively involved in a project, is affected by its process or outcome, or can influence its process or outcome.

Quelle: Wiegers, K., Beatty, J., Software Requirements, 3rd ed. 2013

### *Outside the Developing Organization*

Direct user	Business management	Consultant
Indirect user	Contracting officer	Compliance auditor
Acquirer	Government agency	Certifier
Procurement staff	Subject matter expert	Regulatory body
Legal staff	Program manager	Software supplier
Contractor	Beta tester	Materials supplier
Subcontractor	General public	Venture capitalist

### *Developing Organization*

Development manager	Sales staff	Executive sponsor
Marketing	Installer	Project management office
Operational support staff	Maintainer	Manufacturing
Legal staff	Program manager	Training staff
Information architect	Usability expert	Portfolio architect
Company owner	Subject matter expert	Infrastructure support staff

### *Project Team*

Project manager	Tester
Business analyst	Product manager
Application architect	Quality assurance staff
Designer	Documentation writer
Developer	Database administrator
Product owner	Hardware engineer
Data modeler	Infrastructure analyst
Process analyst	Business solutions architect

Quelle: Wiegers, K., Beatty, J., Software Requirements, 3rd ed. 2013

## Voruntersuchung bzw. Durchführbarkeitsstudie

- Zeigen der fachlichen, ökonomischen und personellen Durchführbarkeit
- Am Ende der Planungsphase steht die Entscheidung über die weitere Vorgehensweise:
  - Weitermachen oder beenden (stop or go)

Quelle: Balzert, H.; Lehrbuch der Softwaretechnik, 2001

## Durchführbarkeitsuntersuchung

- Prüfen der fachlichen Durchführbarkeit
  - Softwaretechnische Realisierbarkeit
  - Verfügbarkeit Entwicklungs- und Zielmaschinen
- Prüfen alternativer Lösungsvorschläge
  - Beispiel: Kauf und Anpassung von Standardsoftware vs. Individualentwicklung
- Prüfen der personellen Durchführbarkeit
  - Verfügbarkeit qualifizierter Fachkräfte für die Entwicklung
- Prüfen der Risiken
- Aufwands- und Termschätzung
- Wirtschaftlichkeitsrechnung

Quelle: Balzert, H.; Lehrbuch der Softwaretechnik, 2001

## Ergebnisse Planungsphase

- Durchführbarkeitsstudie (feasibility study)
- Lastenheft (grobes Pflichtenheft)
- Projektkalkulation
- Projektplan

Benötigte Rollen (M=Mitwirkende, V=Verantwortliche)

Aktivität	Auftraggeber	Projektleiter	Anwendungsspezialist
Lastenheft erstellen	M	M	V
Glossar erstellen	M	M	V
Projektkalkulation erstellen	M	V	M (Aufwand)
Projektplan erstellen	M	V	M

Quelle: Balzert, H.; Lehrbuch der Softwaretechnik, 2001

## Lastenheft

„Vom Auftraggeber festgelegte Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragnehmers innerhalb eines Auftrags“

DIN 69905

Zusammenfassung aller **fachlichen Basisanforderungen**, die das zu entwickelnde Software-Produkt **aus der Sicht des Auftraggebers** erfüllen muss.

Ziel ist die inhaltliche Konkretisierung der gewöhnlich vagen Vorstellungen des Auftraggebers. Dazu muss sich der Systemanalytiker zunächst in den Problembereich einarbeiten und bei der **Festlegung der gemeinsamen Begriffswelt** mitwirken. Er muss in der Lage sein, die Situation aus der Sicht des Auftraggebers zu beurteilen und dessen Probleme sowie Lösungsvorschläge nachzuvollziehen.

Im Mittelpunkt steht nicht so sehr die Präzisierung der Funktionalität bis ins letzte Detail, sondern die **Erlangung eines Gesamtüberblicks**.

Das Lastenheft wird in **natürlicher Sprache** abgefasst und richtet sich an die Planer und Entscheidungsträger von Auftraggeber und Auftragnehmer.

Das Lastenheft wird auch **grobes Pflichtenheft** genannt.

## Lastenheftgliederung

### 1. Visionen und Ziele

Beschreibung der Ziele, die durch den Einsatz des Produktes erreicht werden sollen.

### 2. Rahmenbedingungen

Festlegung, für welche Anwendungsbereiche und für welche Zielgruppen das Produkt vorgesehen ist.

### 3. Kontext im Überblick

Darstellung mit welchen angrenzenden Systemen das zukünftige Softwareprodukt über Schnittstellen interagiert.

### 4. Funktionale Anforderungen

Beschreibung der Hauptfunktionen des Produktes aus Auftraggebersicht. Es werden nur Kernfunktionen beschrieben, auf Detailbeschreibungen wird verzichtet.

### 5. Qualitätsanforderungen

Auflistung der wichtigsten Qualitätsanforderungen

## Lastenheftgliederung (DIN 69905)

- Zielsetzung
- Produkteinsatz
- Produktübersicht
- Funktionale Anforderungen
- Nicht funktionale Anforderungen
- Risikoakzeptanz
- Skizze des Entwicklungszyklus
- Lieferumfang
- Abnahmekriterien

## Methoden

### Outside-in-Methode

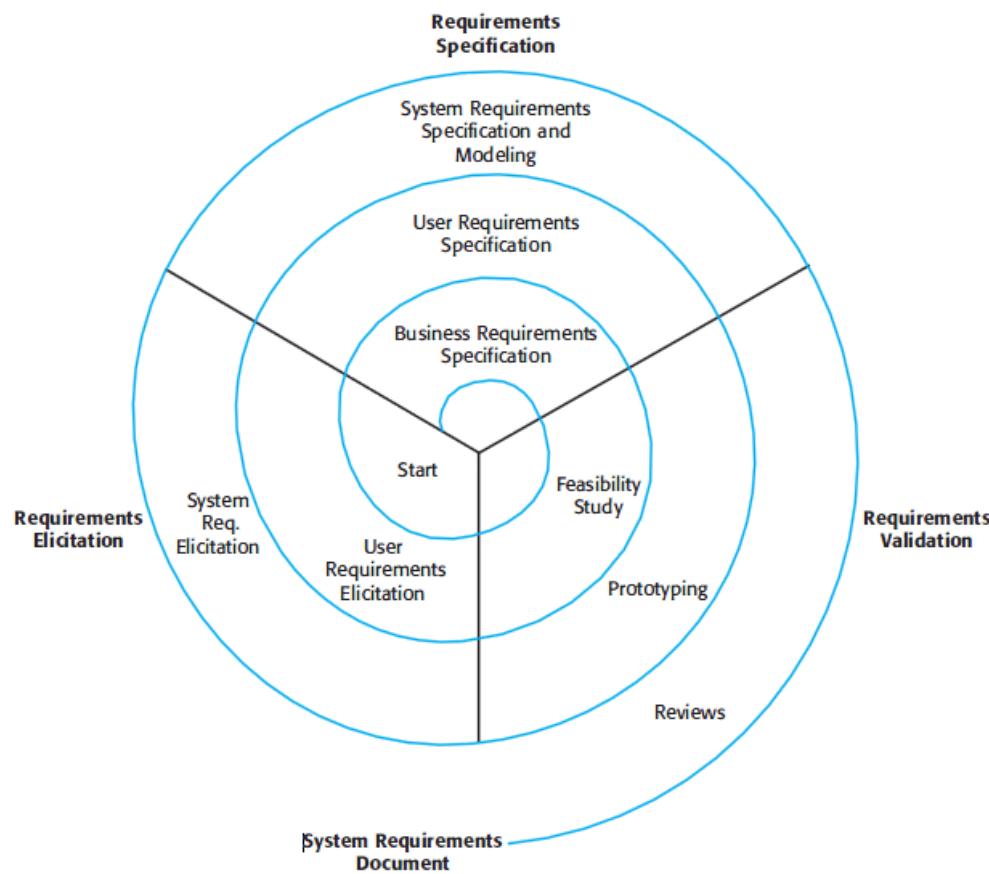
- Zunächst wird die Umwelt eines Produktes modelliert und davon ausgehend die Produktinterna

### Inside-out-Methode

- Zunächst werden die Produktinterna und dann die Schnittstellen zur Umwelt modelliert

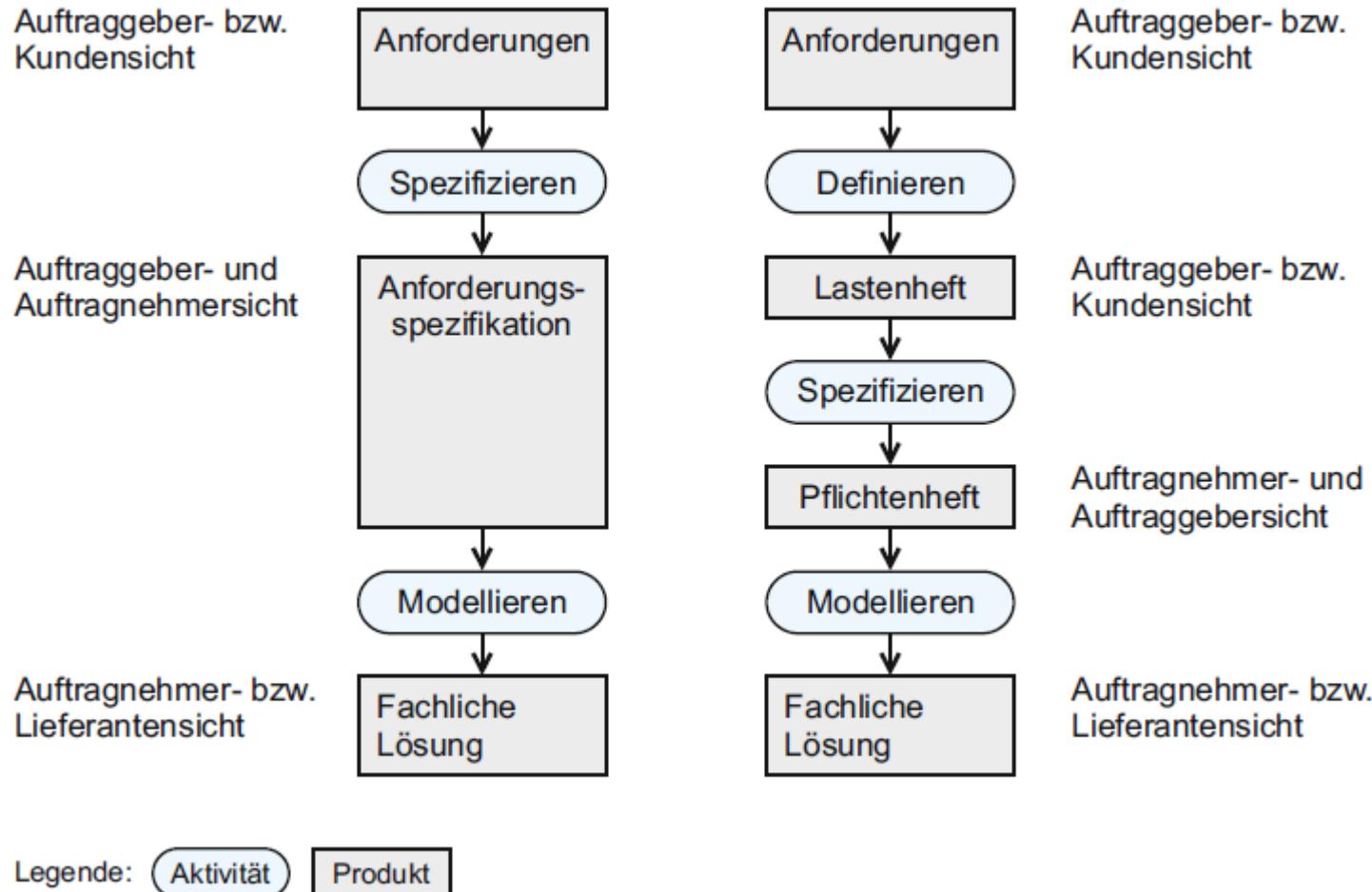
Quelle: Balzert, H.; Lehrbuch der Softwaretechnik, 2001

## Anforderungsarten



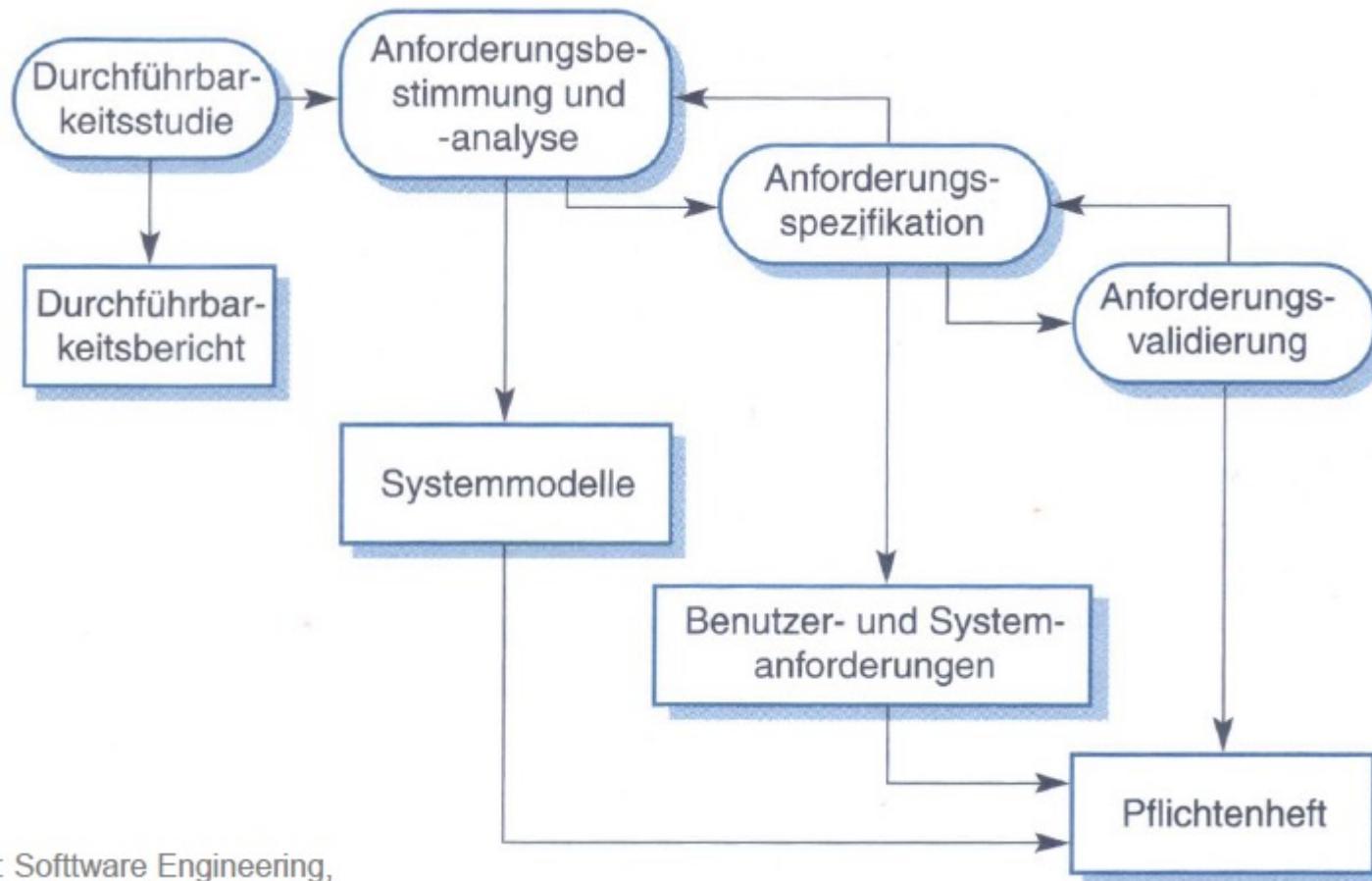
Quelle: Sommerville, I.; Software Engineering, 9th ed.. 2011

## Anforderungen – Aktivitäten und Artefakte



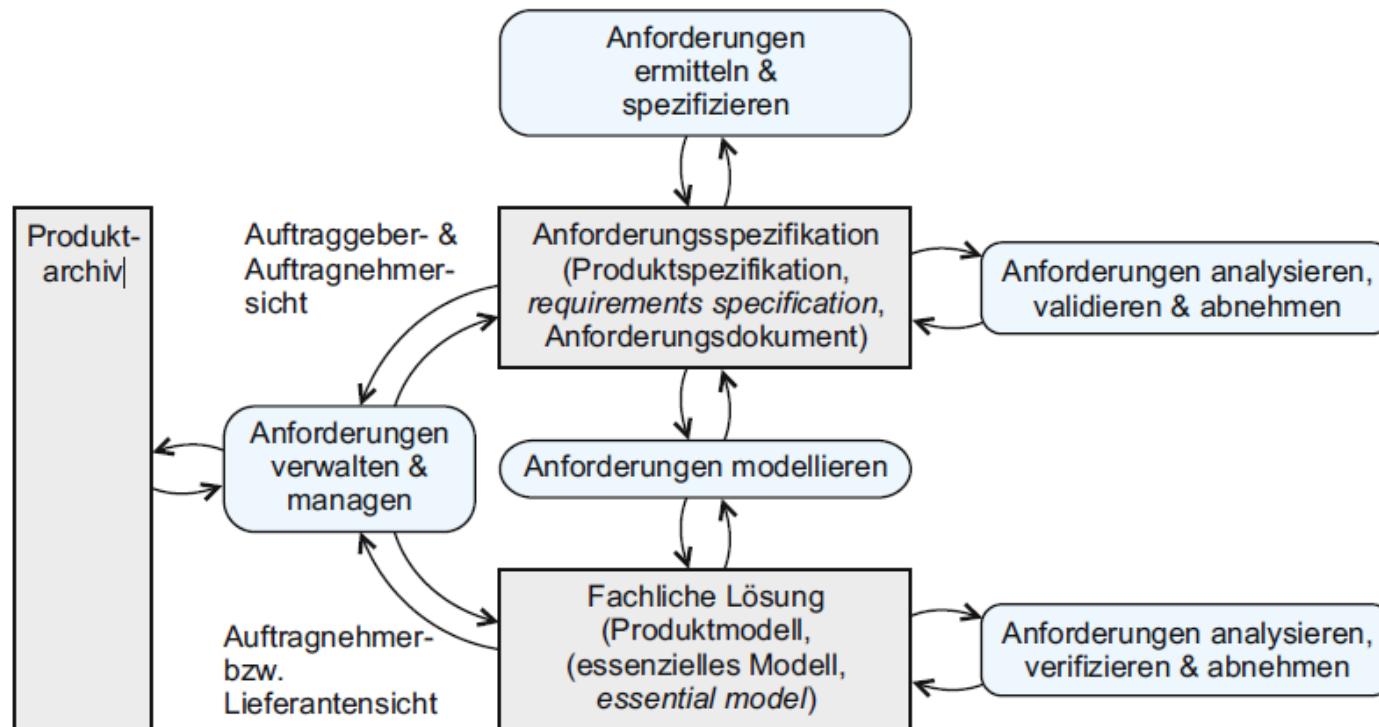
Quelle: Balzert, H.; Softwaretechnik – Basiskonzepte und Requirements Engineering, 2009

## Anforderungen – Aktivitäten und Artefakte



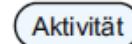
Quelle: Software Engineering,  
Ian Sommerville, 2001

## Anforderungen –Aktivitäten und Artefakte



Legende:

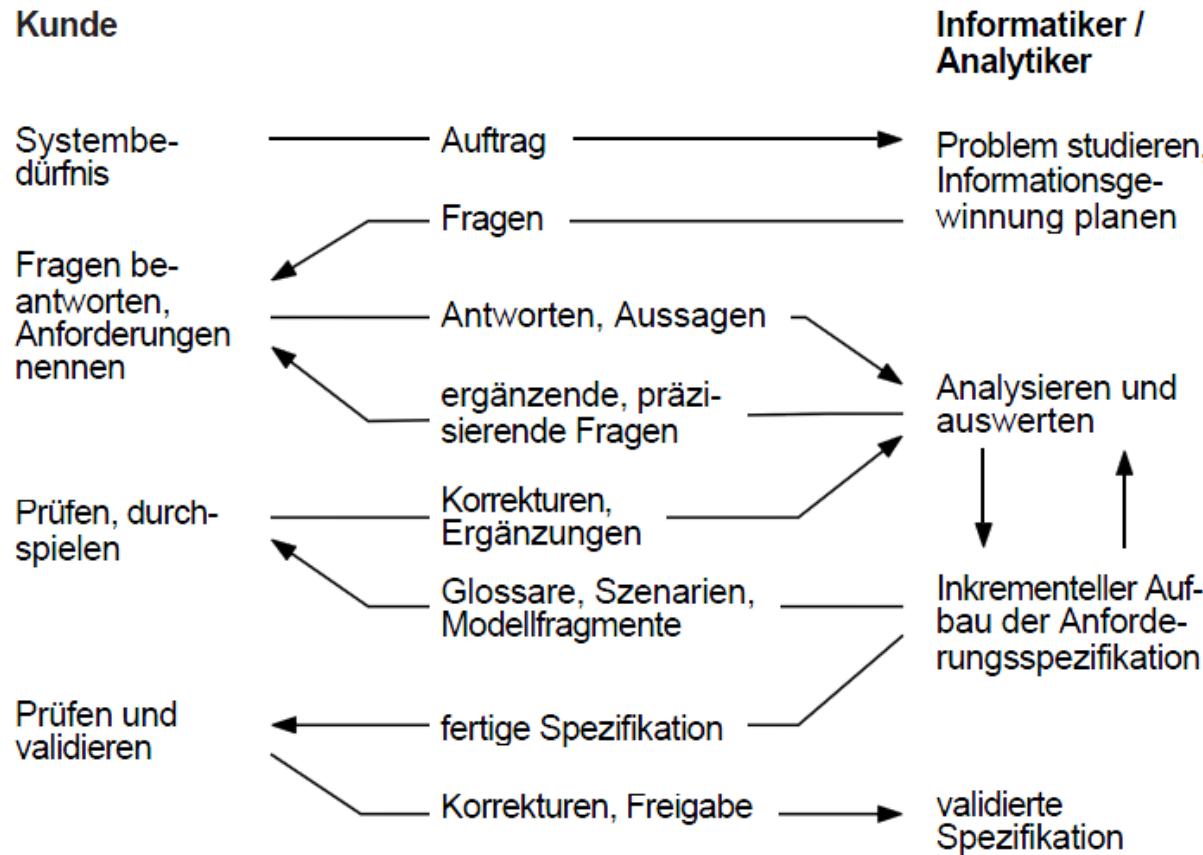
 Informationsfluss

 Aktivität

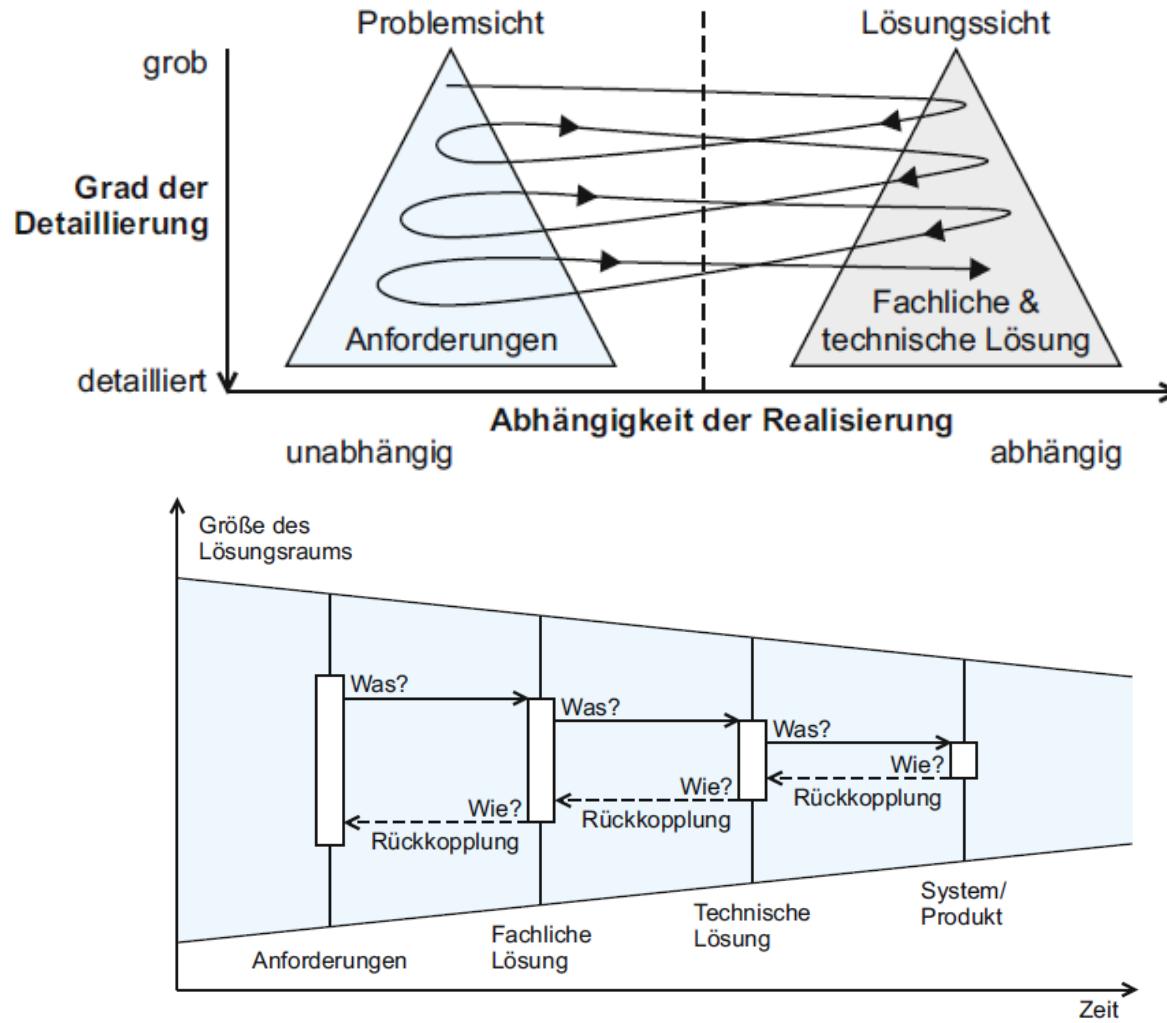
 Artefakt

Quelle: Balzert, H.; Softwaretechnik – Basiskonzepte und Requirements Engineering, 2009

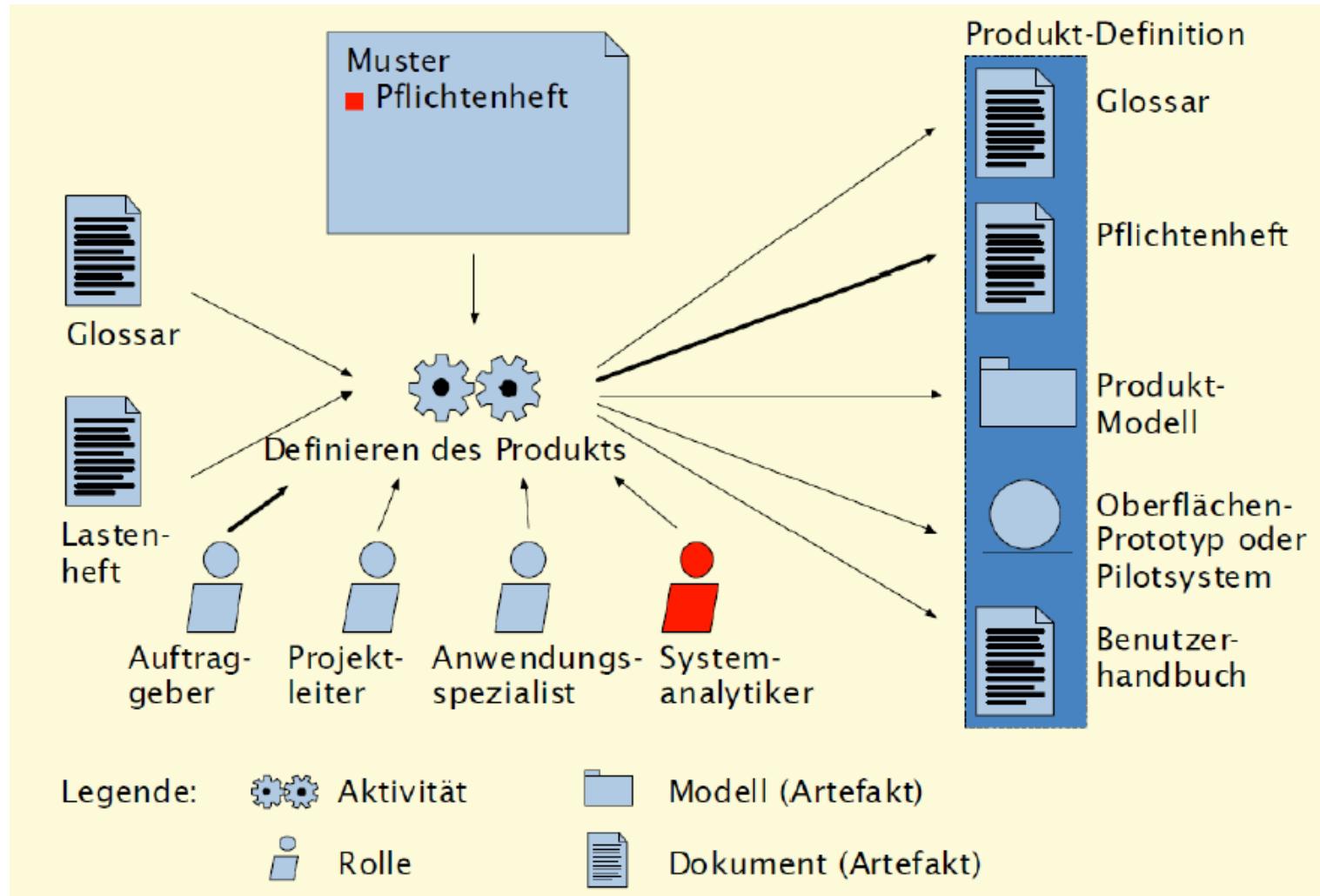
## Wechselspiel Kunde - Analytiker



## Wechselspiel Anforderungen – fachl./techn. Lösung



Quelle: Balzert, H.; Softwaretechnik – Basiskonzepte und Requirements Engineering, 2009



## Pflichtenheft

„Beschreibung der Realisierung aller Anforderungen des Lastenheftes. Das Pflichtenheft enthält das Lastenheft. Im Pflichtenheft werden die Anwendervorgaben detailliert und die Realisierungsanforderungen beschrieben. Im Pflichtenheft wird definiert, WIE und Womit die Anforderungen zu realisieren sind.[...]

Das Pflichtenheft wird in der Regel nach Auftragserteilung vom Auftragnehmer erstellt, falls erforderlich unter Mitwirkung des Auftraggebers. Der Auftragnehmer prüft bei der Erstellung des Pflichtenhefts die Widerspruchsfreiheit und Realisierbarkeit der im Lastenheft genannten Anforderungen. Das Pflichtenheft bedarf der Genehmigung durch den Auftraggeber.“

VDI 2519

„Vom Auftragnehmer erarbeitete Realisierungsvorgaben aufgrund der Umsetzung des vom Auftraggeber vorgegebenen Lastenheftes“

DIN 69905

Quelle: Balzert, H.; Softwaretechnik – Basiskonzepte und Requirements Engineering, 2009

## Gliederung Pflichtenheft

### 1. Visionen und Ziele

Ziele des zu entwickelndes Systems (Muss-, Wunsch- uns Abgrenzungskriterien)

### 2. Rahmenbedingungen

Der Anwendungsbereich und die Akteure des zukünftigen Systems werden genannt

### 3. Kontext im Überblick

Darstellung angrenzender Systeme an das zukünftigen Softwaresystem

### 4. Funktionale Anforderungen

Detaillierte Beschreibung der Anwendungsfälle und Daten des Systems

### 5. Qualitätsanforderungen

Gewichtung der nicht-funktionalen Anforderungen in einer Tabelle

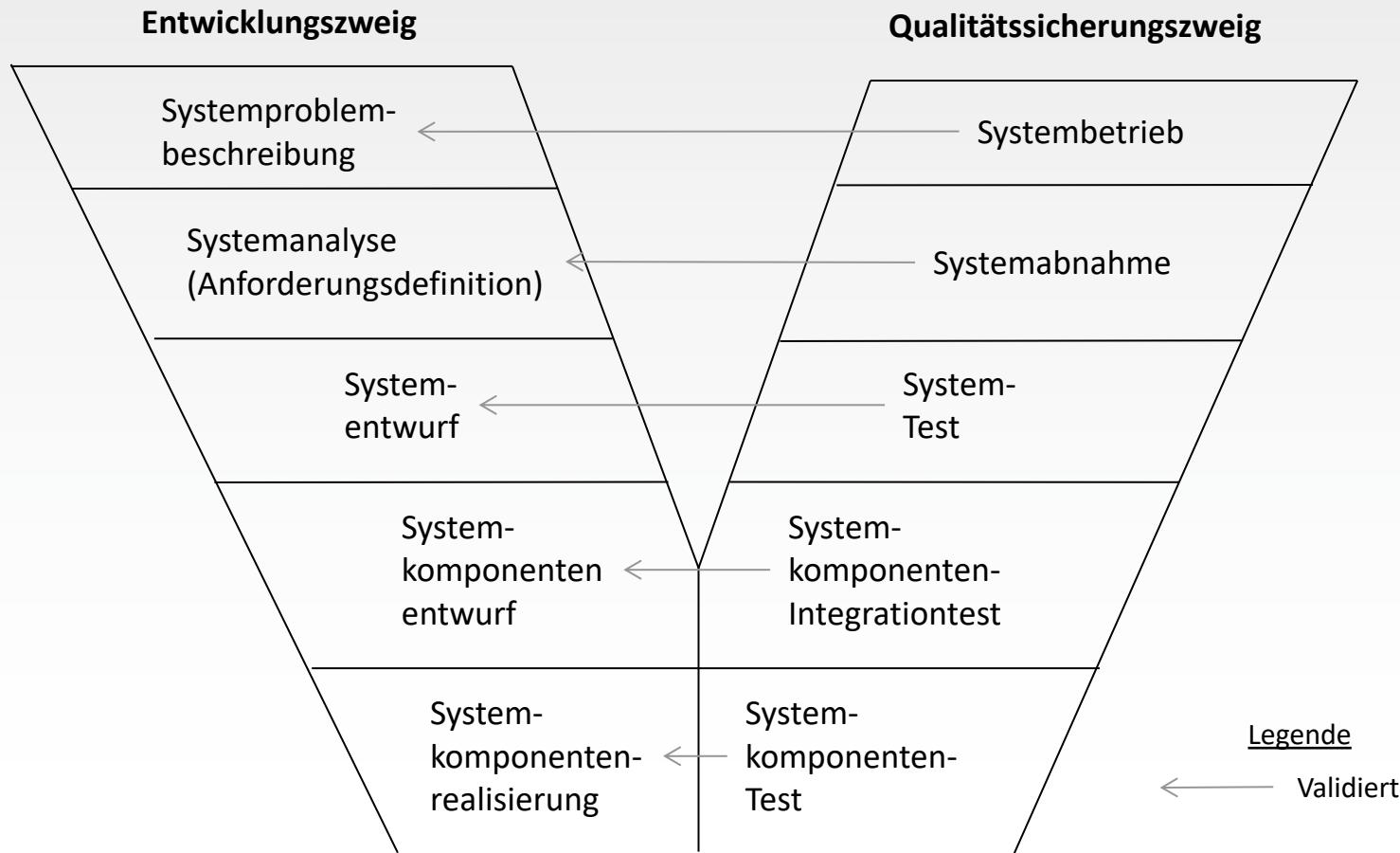
### 8. Abnahmekriterien

Nennung von Kriterien, die bei der Systemabnahme kontrolliert werden

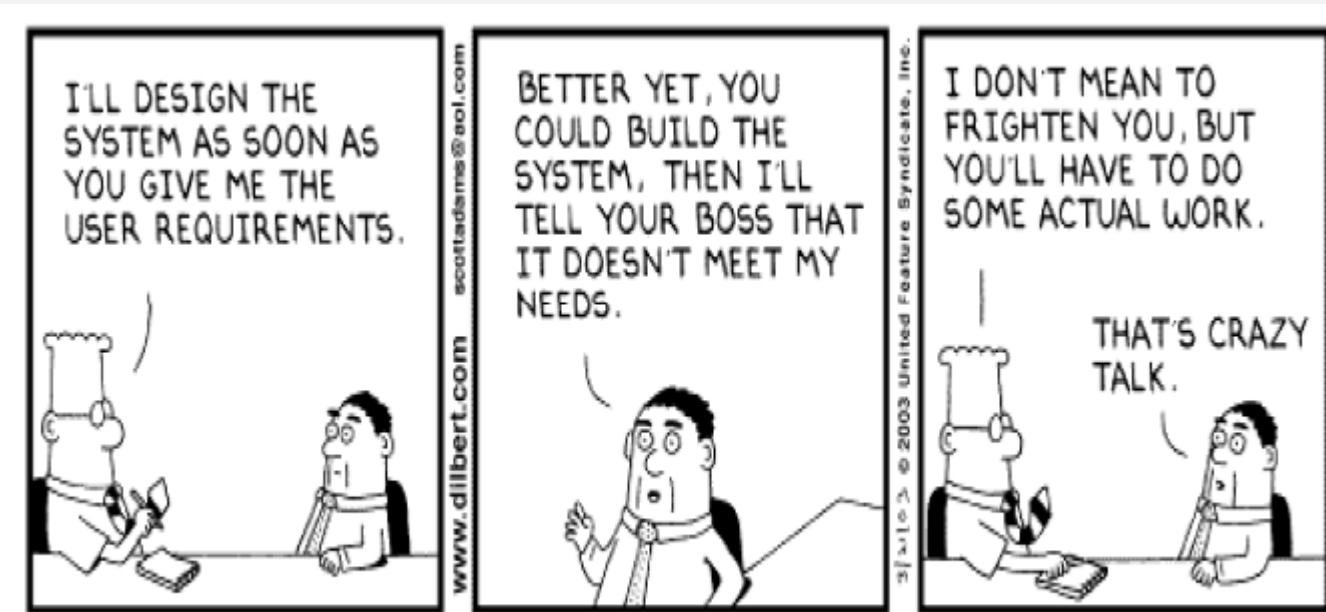
### 9. Subsystemstruktur (optional)

Beschreibung von Subsystemen des zukünftigen Softwaresystems

## V-Modell

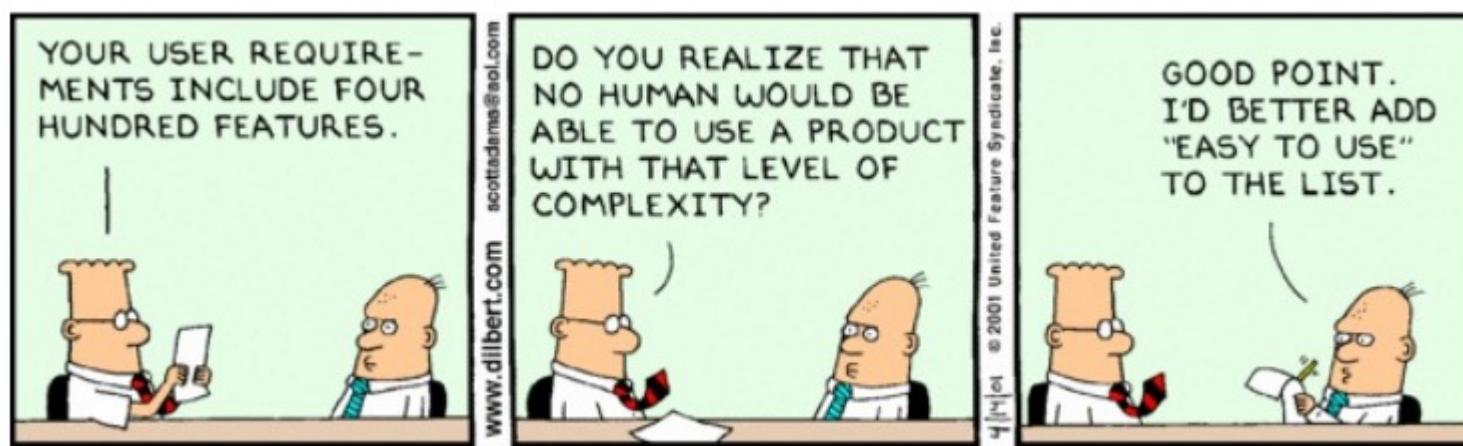


## Anforderungen



Copyright © 2003 United Feature Syndicate, Inc.

## Anforderungen



# **Computer Aided Software Engineering (CASE)**

## Definition

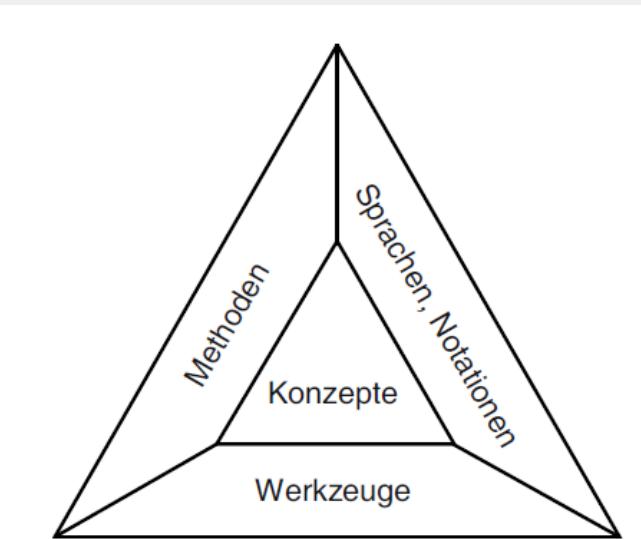
„Abk. für *Computer Aided Software Engineering*; über den gesamten Entwicklungsprozess von Software bereitgestellte Computerunterstützung bez. Methoden und Werkzeugen. Die Unterstützung erfolgt über sog. CASE-Tools, die meist folgende Eigenschaften besitzen: Einheitliches Data Dictionary (DD) als Grundlage, grafische Entwicklungsoberfläche, automatische Generierung von Programmcodes und Unterstützung der Dokumentation und des Projektmanagements. Unter einer einheitlichen Oberfläche werden für jede Phase des Entwicklungsprozesses unterschiedliche Werkzeuge angeboten, die die Entwickler in ihrer Teamarbeit unterstützen. Eine wichtige Anforderung an CASE-Tools besteht aufgrund der enormen Dynamik des IT-Bereichs in der Erweiterbarkeit um weitere Methoden und der Unterstützung mehrerer grundlegender Systeme.“

Quelle: <http://wirtschaftslexikon.gabler.de/Archiv/75156/case-v7.html> (Zugriff: 11.10.2013)

„The use of computers to aid in the software engineering process. May include the application of software tools to software design, requirements tracing, code production, testing, document generation, and other software engineering activities.“

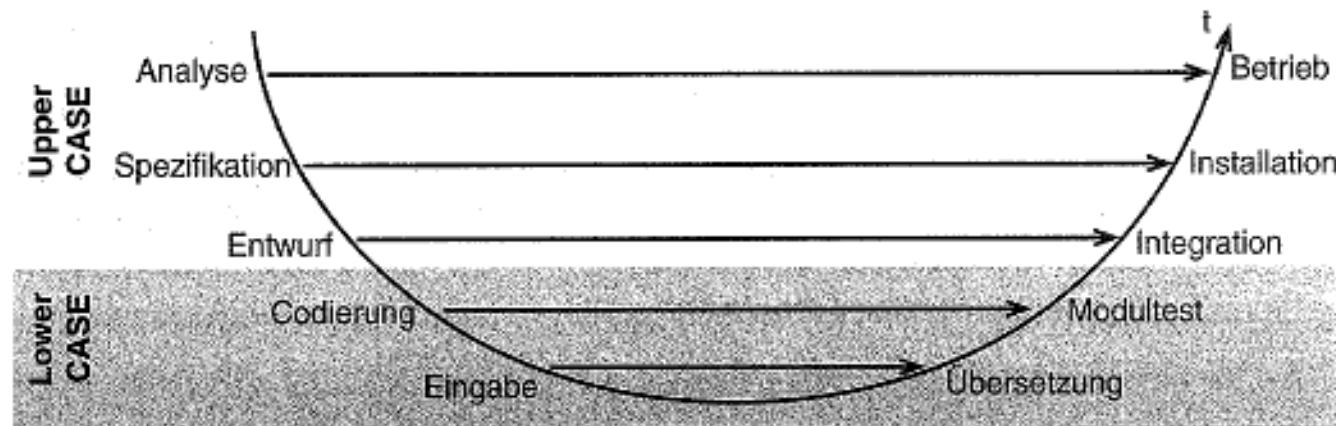
Quelle: IEEE Std 610.12 (1990)

„Ein **Werkzeug** dient zur Ausführung einer Arbeit, die – wenigstens prinzipiell – auch ohne das Werkzeug geleistet werden könnte. Im Produkt ist das Werkzeug nicht mehr enthalten, es sei denn als Ausrüstung für die Wartung“



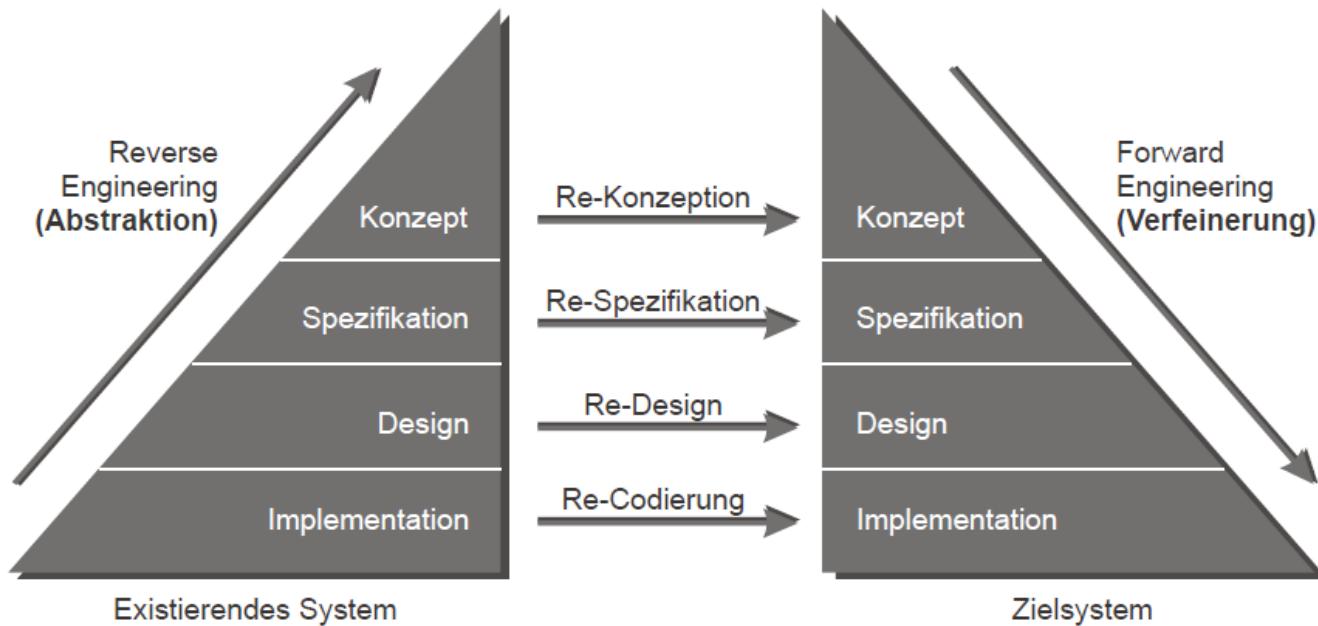
Quelle: Ludewig und Lichter, Software Engineering, 2010

## Klassifikation



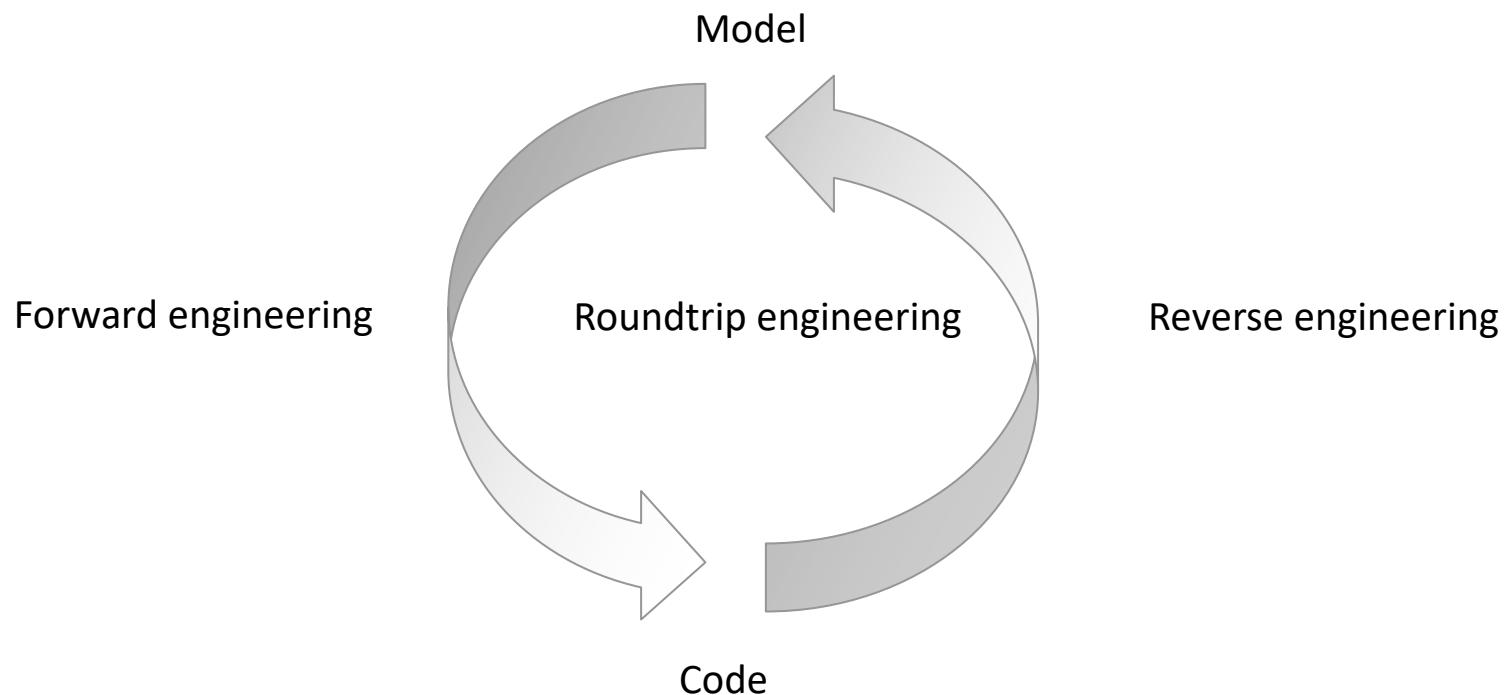
Quelle: Ludewig und Lichter, Software Engineering, 2010

## Forward-/Reverse-Engineering

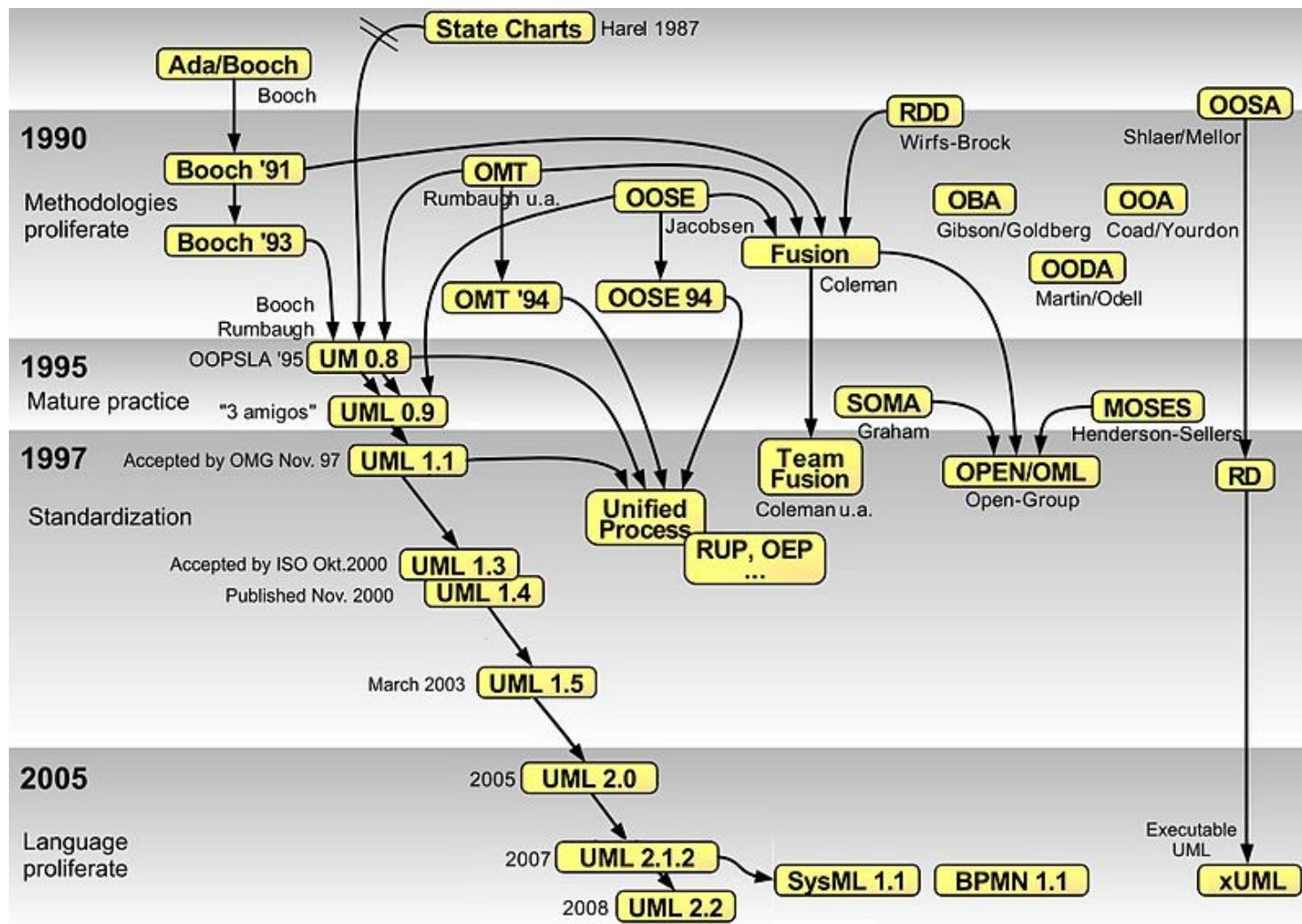


Quelle: nach Byrne, E.; A Conceptual Foundation for Software Re-Engineering, 1992

## Roundtrip Engineering



# **Unified Modeling Language (UML)**



Quelle: [http://en.wikipedia.org/wiki/File:OO\\_Modeling\\_languages\\_history.jpg](http://en.wikipedia.org/wiki/File:OO_Modeling_languages_history.jpg) (Zugriff: 11.10.2013)

## Definition

**„The OMG's Unified Modeling Language™ (UML®) helps you specify, visualize, and document models of software systems ....“**

Quelle: [http://www.omg.org/gettingstarted/what\\_is.uml.htm](http://www.omg.org/gettingstarted/what_is.uml.htm) (Zugriff: 11.10.2013)

**„Die UML ist in erster Linie die Beschreibung einer einheitlichen Notation und Semantik sowie die Definition eines Metamodells“.**

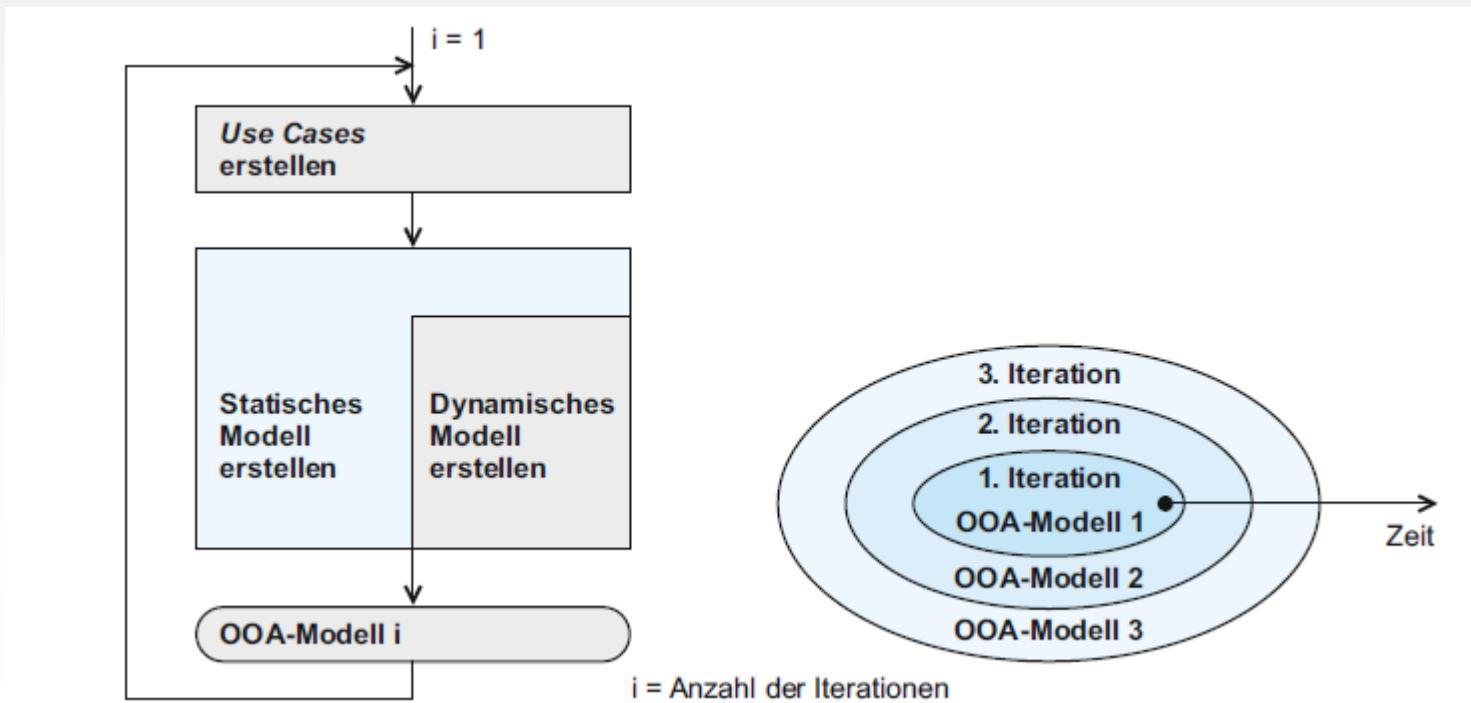
Quelle: Oesterreich, 2012

**„The goal of the UML is to provide a standard notation that can be used by all object-oriented methods and to select and integrate the best elements of precursor notation“**

Quelle: Bruegge, 2010

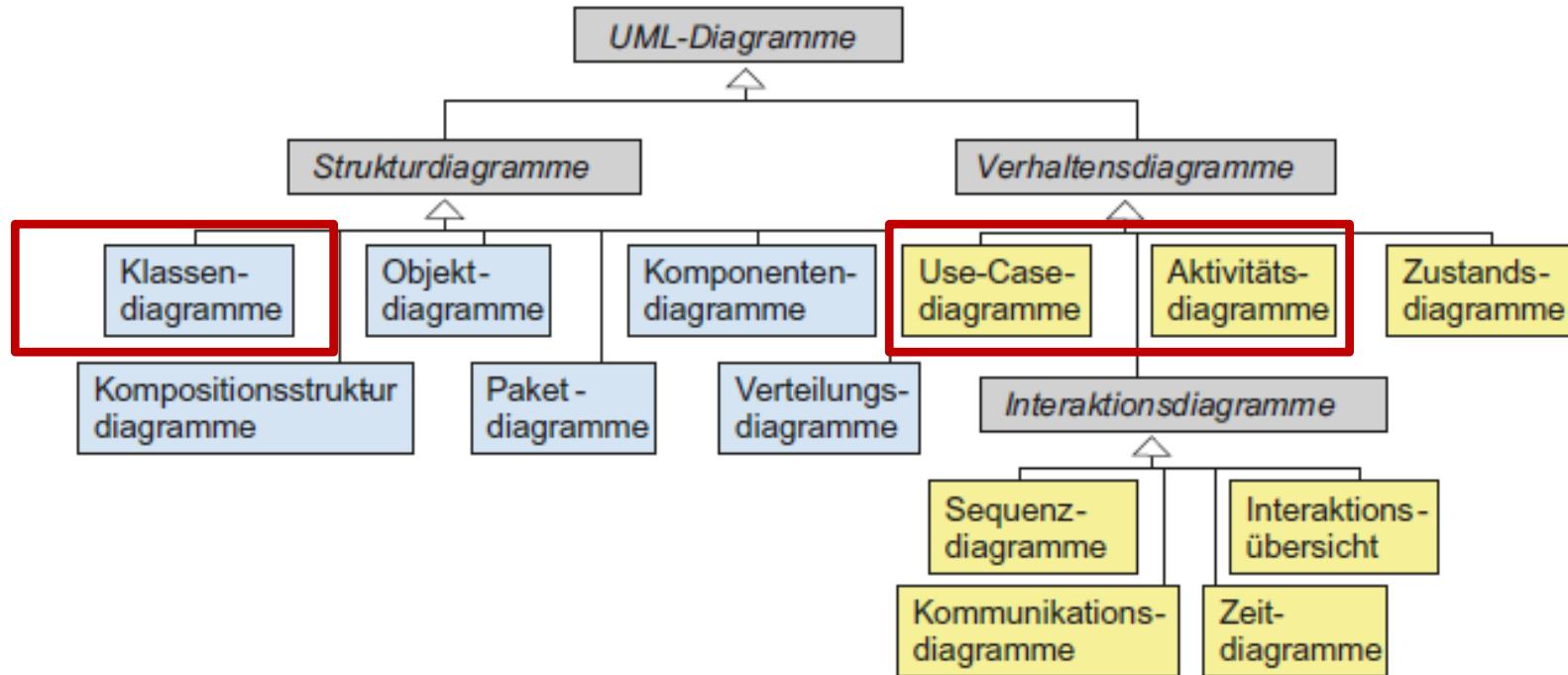
„Der Makroprozess berücksichtigt die Gleichgewichtigkeit (*balancing*) von statischem und dynamischem Modell (balancierter Makroprozess).“

Quelle: Balzert, H.; Lehrbuch der Softwaretechnik – Basiskonzepte und Requirements Engineering, 2009

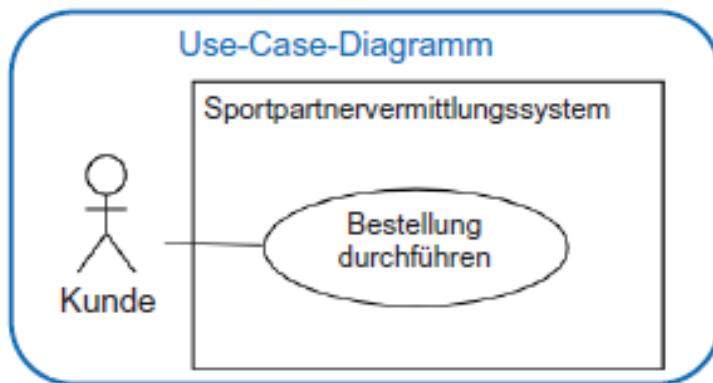


# **Unified Modeling Language (UML) - Diagramme**

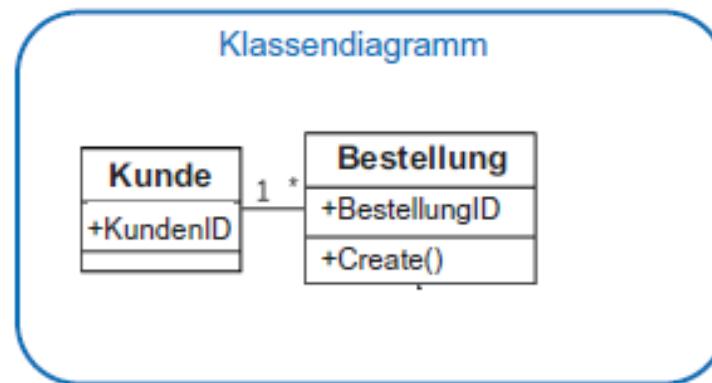
Legende  
In Vorlesung  
behandelt



## Verhaltensdiagramm



## Strukturdiagramm



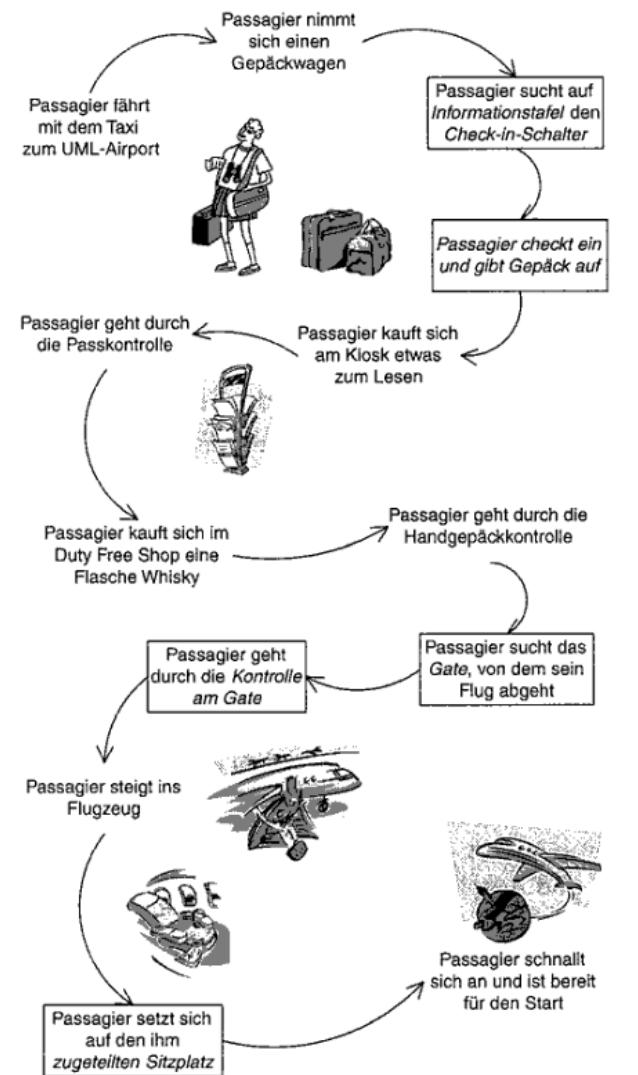
# **Unified Modeling Language (UML) – Anwendungsfalldiagramm (Use Case diagram)**

## Aufgabe / Eigenschaften

- Funktionen werden aufgelistet , die das Anwendungssystem bereitstellt (funktionale Black-Box-Sicht)
- Nutzer des Anwendungssystems werden in Form von Akteuren dargestellt
- Funktionen werden aus Nutzersicht definiert
- Ein Use-Case-Diagramm sollte nicht mehr als 10 Use-Cases enthalten
- Dynamische Sicht des Anwendungssystems

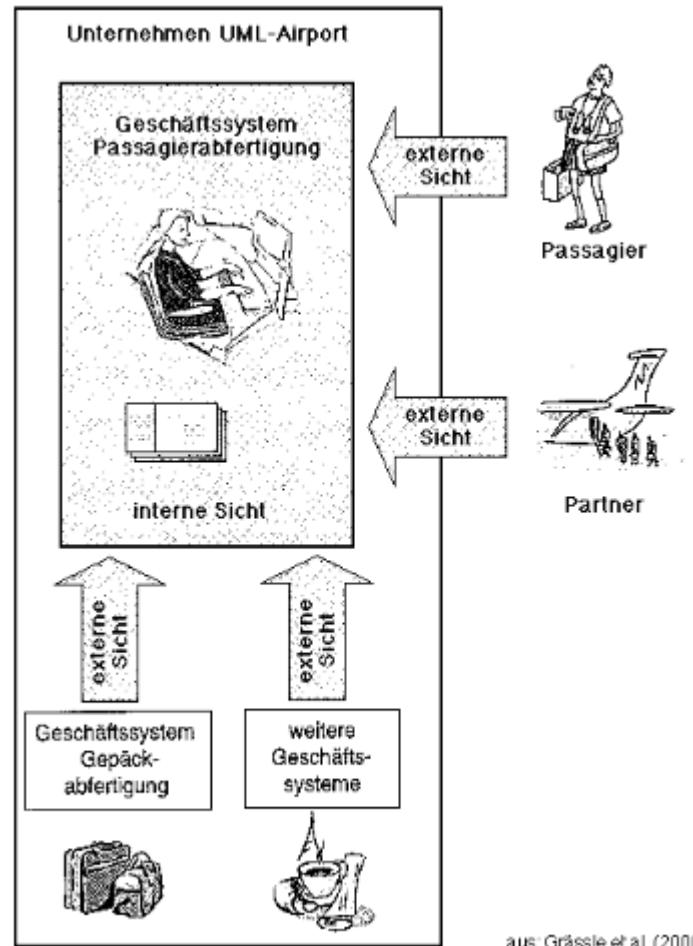
# Use Case

„Check-in passenger“



aus: Gräßle et al. (2000)

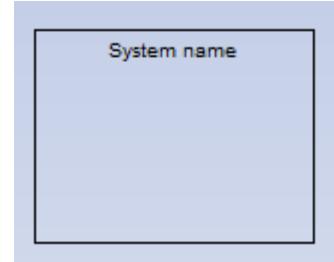
„Check-in passenger“



aus: Gräßle et al. (2000)

## System (system)

Stellt das zu analysierende System einer Anwendungsdomäne dar.



## Akteur (actor)

Repräsentiert eine Rolle in der eine natürliche Person oder ein Hard-/Softwaresystem mit dem Anwendungssystem interagiert.



## Anwendungsfall (use case)

Stellt eine Funktion des Systems dar (Geschäftsprozess), mit der ein Akteur interagiert um ein Ergebnis von Wert zu erzielen.



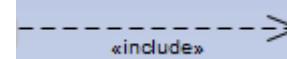
## Assoziation (association)

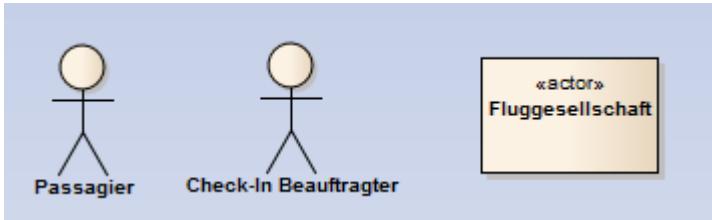
Stellt eine Interaktionsbeziehung zwischen Akteur und dem Anwendungsfall dar.



## Einfüge-Beziehung (include relation)

Stellt eine Beziehung zwischen 2 Anwendungsfällen dar, bei der der Anwendungsfall auf dem mit dem Pfeil verwiesen wird im anderen enthalten ist





## Passagier

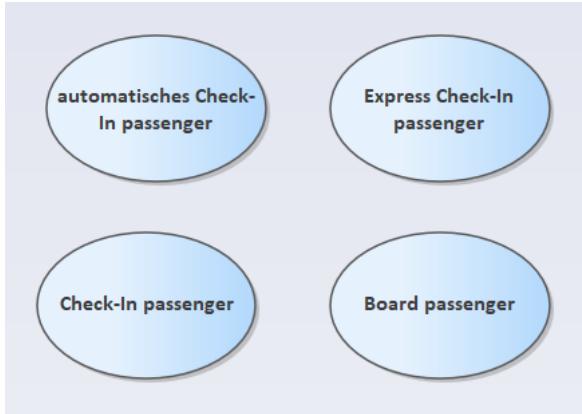
Eine natürliche Person die ein Flugreise gebucht hat und als Fluggast an einem Flug teilnimmt.

## Fluggesellschaft

Unternehmen welches Fluggäste von einem Startflughafen zu einem Zielflughafen mittels Flugzeug befördert.

## Check-In-Beauftragter

Person, die anhand von Ticket und Ausweis das Fluggepäck entgegennimmt und dem Fluggast seine Boarding-Card ausstellt.



## Check-In passenger

Vorlegen des Ticket, Gepäckaufgabe, Platzreservierung,  
Übermittlung der Passagierliste an Fluggesellschaft,  
Erstellen und Aushändigen der Boarding-Card

## Express Check-In passenger

Wird von Passagieren genutzt, die nur Handgepäck haben  
(Keine Gepäckaufgabe). Ansonsten wie Check-In.

## Automatisches Check-In passenger

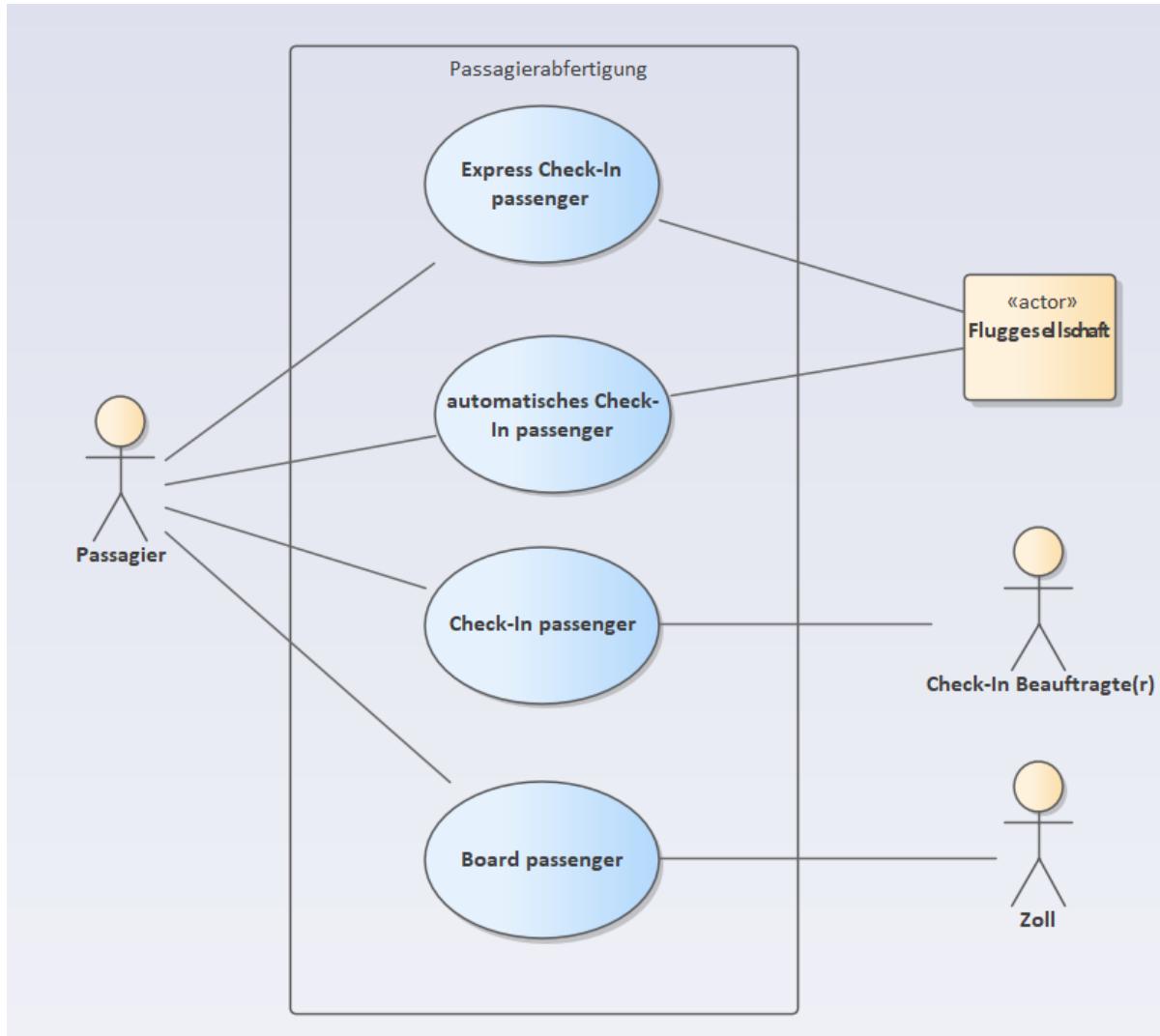
Erfolgt ohne Mithilfe des Check-In-Mitarbeiters direkt am Automaten. Es kann kein Gepäck aufgegeben werden.

## Board passenger

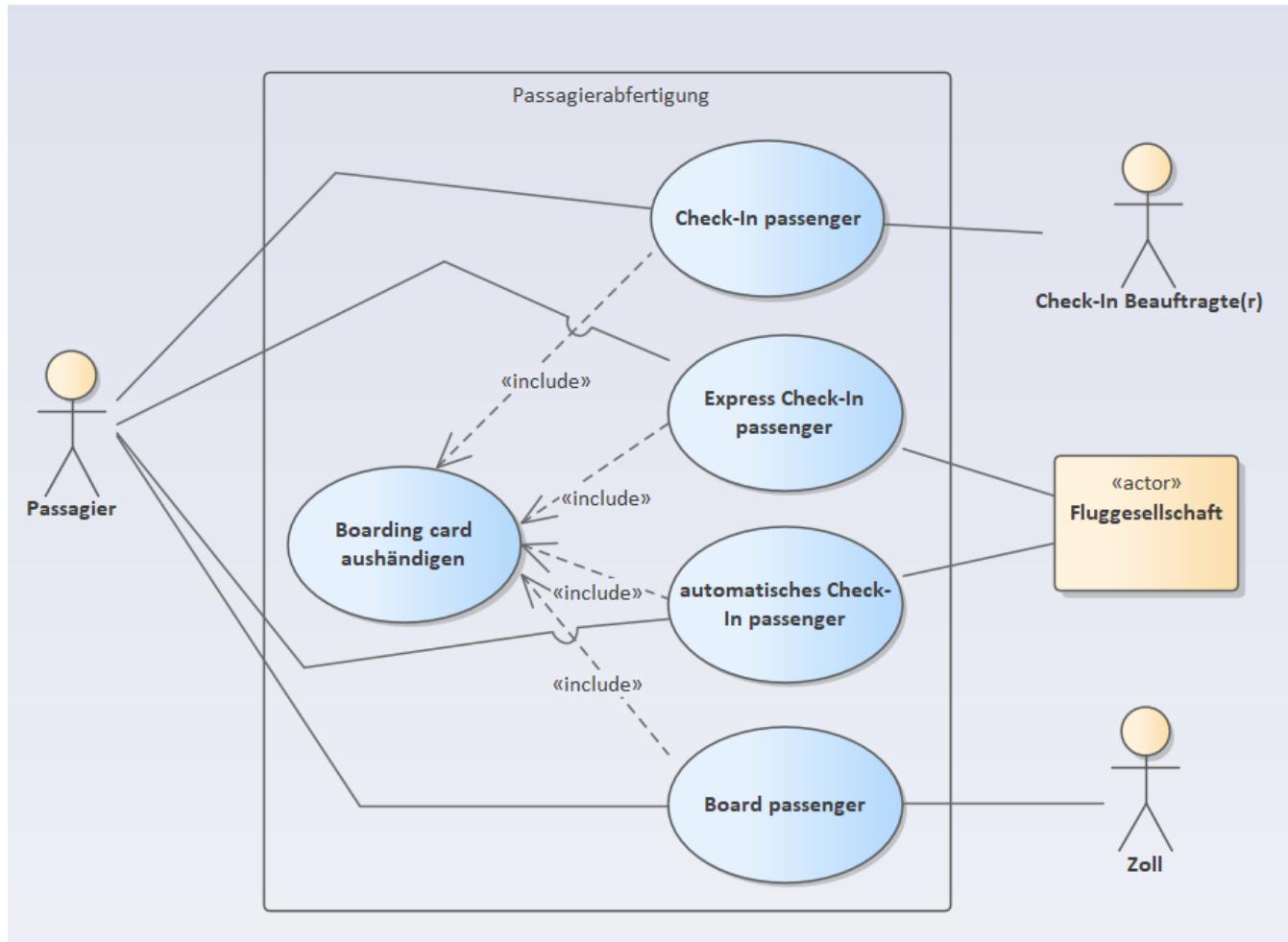
Die Boarding-Card wird durch den Check-In-Beauftragten am Gate kontrolliert.

„Es ist zweckmäßig, Funktionen mit einem Substantiv und einem Verb zu beschreiben. Hierbei ist es wichtig, Verben mit aktivistischer Bedeutung zu verwenden, die das Geschehen direkt beschreiben. Beispielsweise: „Flüssigkeit fördern“ und nicht „Flüssigkeitsförderung ermöglichen“. Das Substantiv soll nach Möglichkeit quantifizierbar sein.“

# Use Case Diagramm - Beispiel



# Use Case Diagramm - Beispiel



## Auslöser

Auslöser eines Anwendungsfalles ist ein Ereignis. Ereignisse können zeitliche Ereignisse, Schwellwertereignisse oder Interaktionsereignisse sein.

Ein zeitliches Ereignis ist das Erreichen eines festgelegten Zeitpunktes (z.B. der 10. eines Monats für die Umsatzsteuererklärung).

Ein Schwellwertereignis ist das Erreichen eines festgelegten mengenmäßigen Wertes (z.B. ist der Mindestbestand für Schrauben von 10.000 erreicht)

Für zeitliche Ereignisse und Schwellwertereignisse wird kein expliziter Akteur im Use Case Diagramm benötigt, da diese implizit durch das Anwendungssystem überprüft und ausgelöst werden.

Ein Interaktionsereignis wird explizit von einem Akteur ausgelöst, da dieser eine Interaktion durch Aufruf der Funktion (Use Case) des Anwendungssystems aufruft. Ein Interaktionsereignis ist z.B. das Aktivieren eines Menueintrages.

## Aktionen

Ein Anwendungsfall besteht aus einer geordneten Menge von einzelnen Aktionen. Nach Abarbeitung der Aktionen liefert der Anwendungsfall ein „Ergebnis von Wert“.

## Ergebnis von Wert

Jeder Anwendungsfall liefert für den Akteur ein Ergebnis von Wert (Leistung), welches weiterverarbeitet werden kann.

## Unabhängigkeit

Die einzelnen Anwendungsfälle im Anwendungsfalldiagramm sind generell unabhängig von einander ausführbar. Ob ein Anwendungsfall ausgeführt werden kann, legen sogenannte Vorbedingungen fest.

# Use Case Beschreibung

**Geschäftsprozess:** Sitzplatz reservieren  
**Kategorie:** primär  
**Vorbedingung:** /LF10/ Kunde angemeldet  
**Nachbedingung Erfolg:** Sitzplatz ist für Fluggast reserviert  
**Nachbedingung Fehlschlag:** Anzeige keine Sitzplätze mehr vorhaben  
**Akteure:** Fluggast  
**Auslösendes Ereignis:** Menupunkt „Sitzplatz reservieren“ wurde aktiviert

Aktivitätsnummer	Akteur (Fluggast)	System (Passagierabfertigung)
1		Der Sitzplatzreservierungsdialog wird angezeigt
2	Es werden Flugdatum, Flugnummer, Startflughafen, Zielflughafen und Name eingegeben	
3		Flug- und Namenseingaben werden überprüft
4		Die Sitzplatzbelegung wird mit einem vorgeschlagenen Sitzplatz angezeigt
5	Der Knopf „Sitzplatz akzeptieren“ wird gedrückt	
6		Der Sitzplatz wird auf der Flugbuchung vermerkt und als „belegt“ bei den Sitzplätzen markiert.

# Use Case Beschreibung

Erweiterung	Akteur (Fluggast)	System (Passagierabfertigung)
3a		[Eingaben falsch] Anzeigen des ausgefüllten Sitzplatzdialoges mit Markierung falschen Eingaben → 2
Alternativen		
5a	[Sitzplatz unerwünscht] Ein anderer Sitzplatz als der vorgeschlagene wird selektiert	

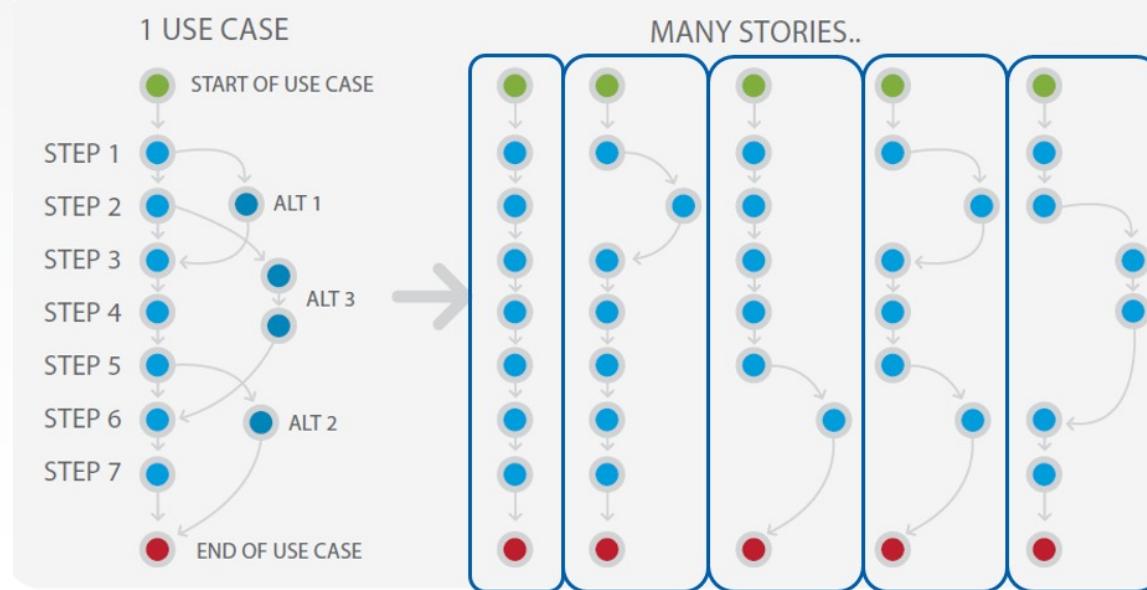
# Use Case vs User story

## Use case

→ Plan-based development

## User story

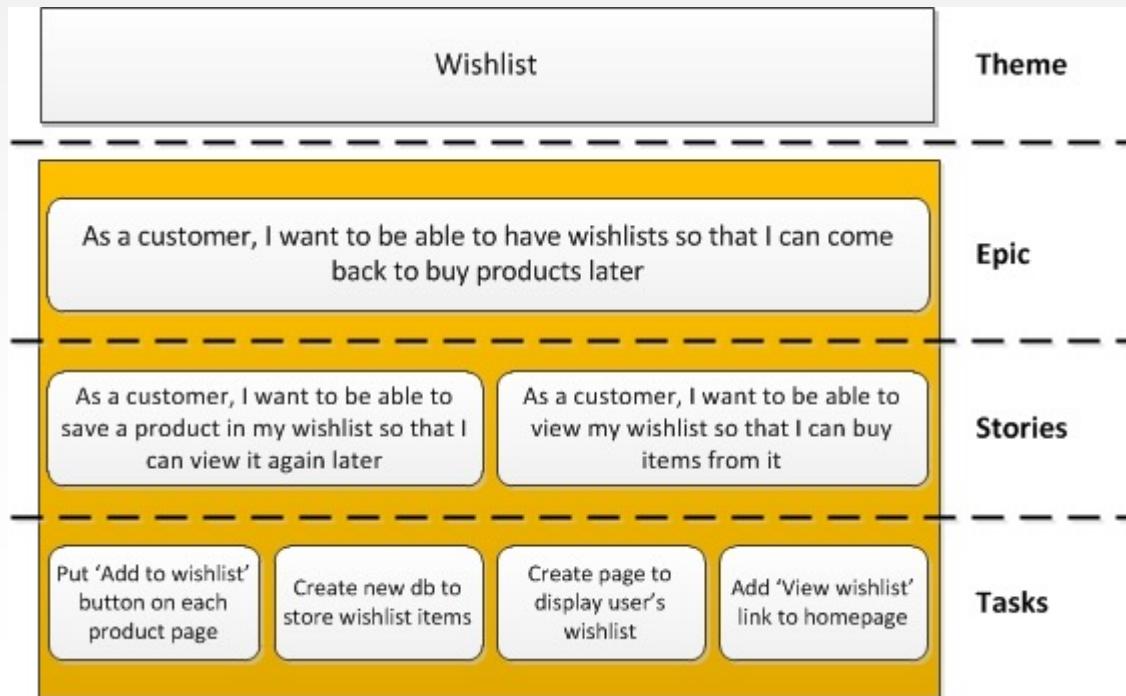
→ Agile development



Quelle: Jacobsen, I.; Use Case 2.0 – The definitive guide, 2011

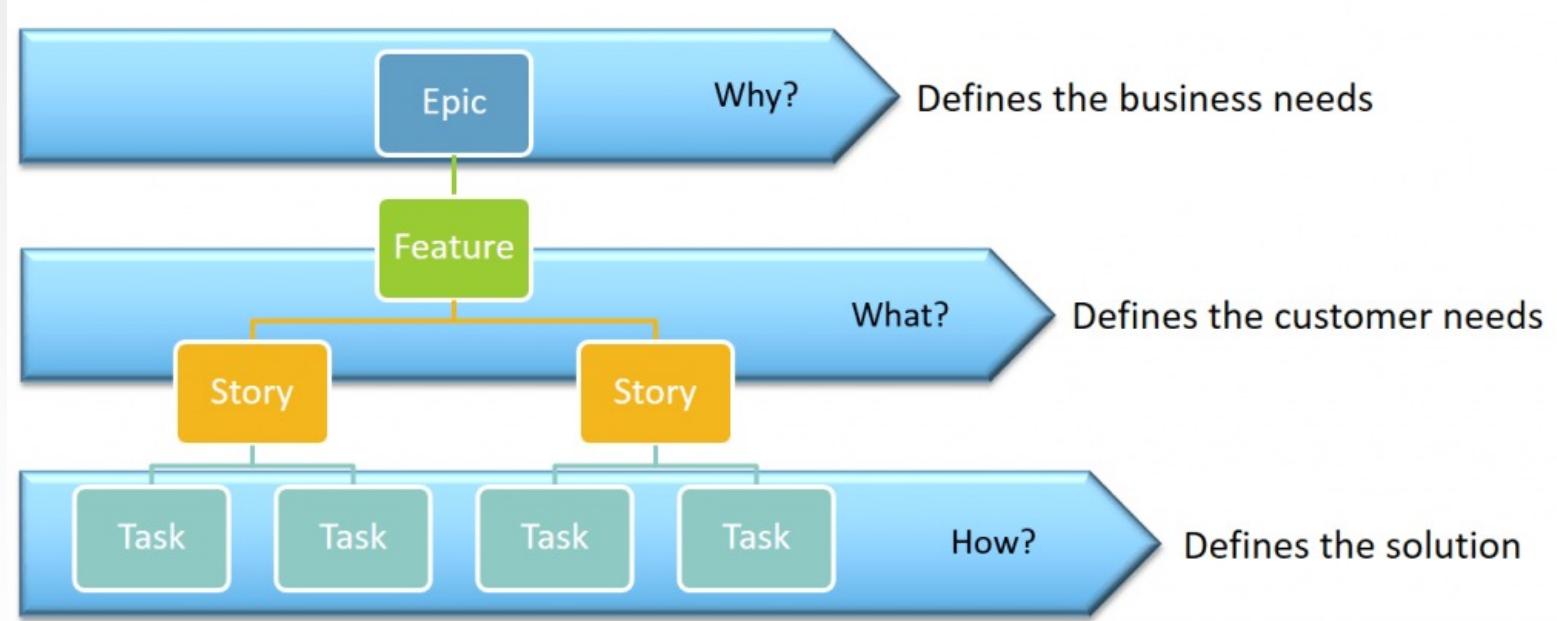
# Theme vs epic vs user story vs task

As a *<user role>* I want *<goal>* so that *<benefit>*



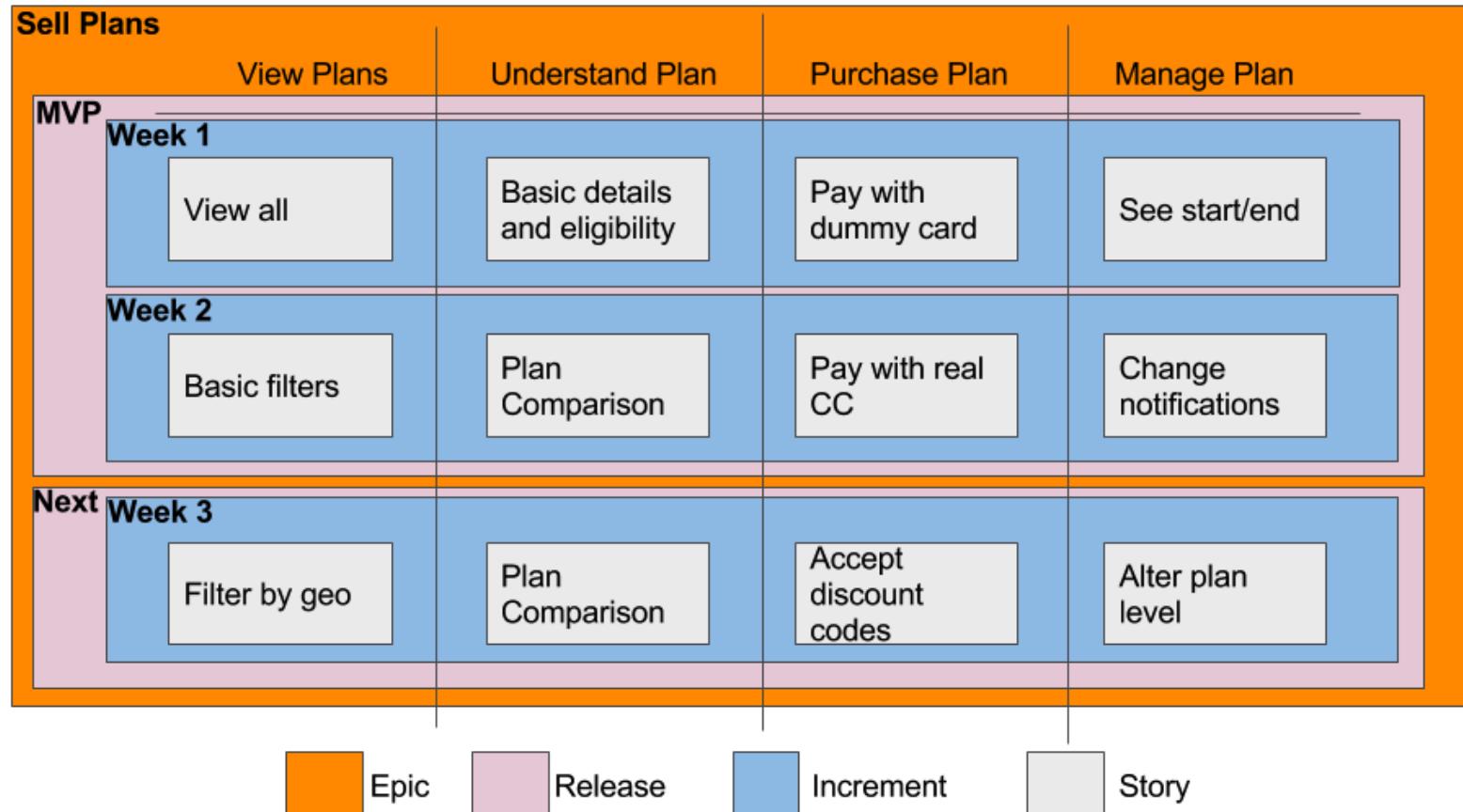
Quelle: <https://scrumandkanban.co.uk/theme-epic-story-task/>; Zugriff: 10.12.2018

# Epic vs user story vs task



Quelle: <https://jthoyer.wordpress.com/experience-design-definitions-and-quotes/agile-epic-feature-story/>; Zugriff: 10.12.2018

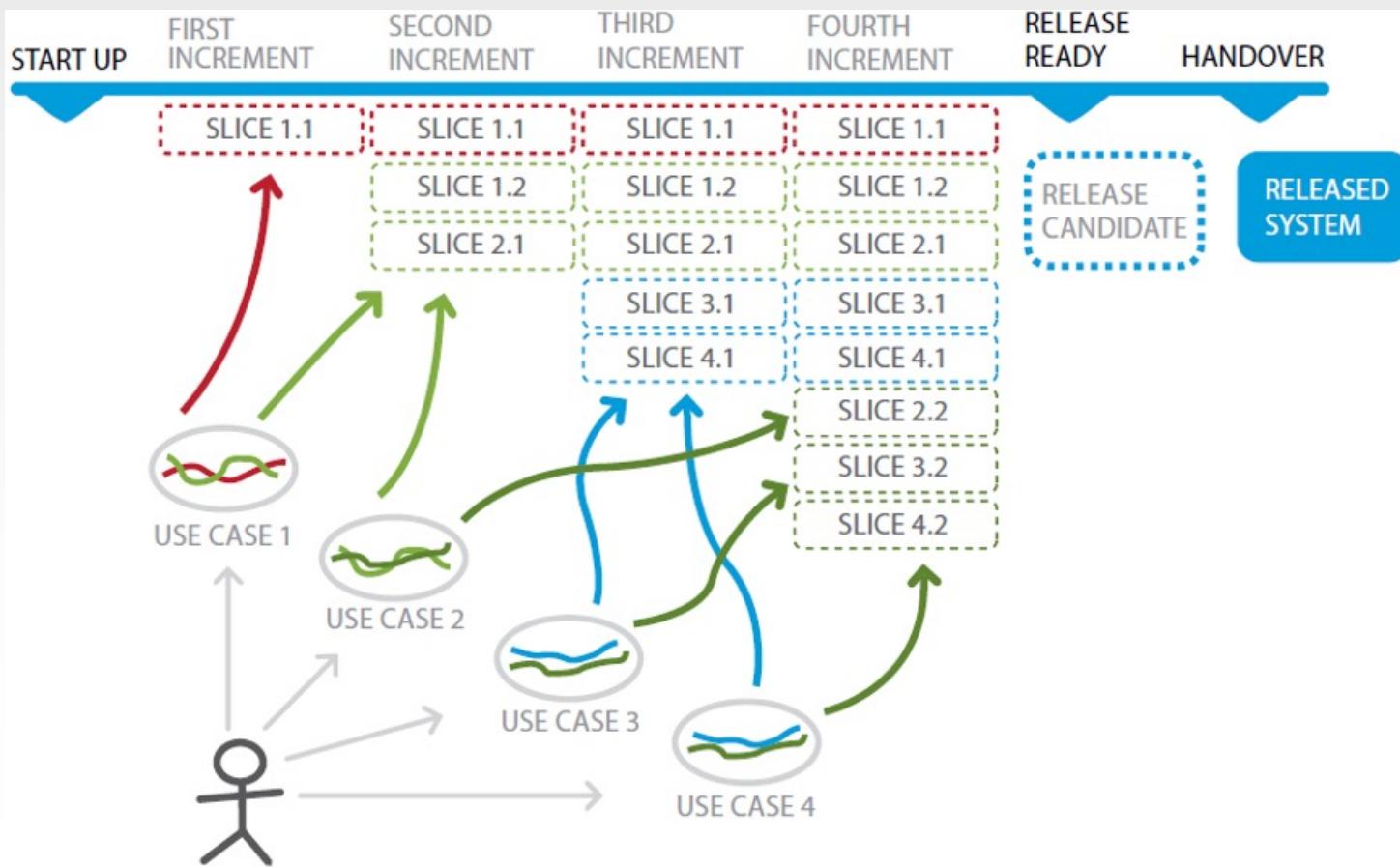
# Epic vs user story vs increment



MVP = Minimum viable product

Quelle: <https://hackernoon.com/stories-vs-epics-d773118420d2>; Zugriff: 10.12.2018

# Use Case vs User story



Quelle: Jacobsen, I.; Use Case 2.0 – The definitive guide, 2011

# Plan-based vs agile-based

	Plan-based	Agile-based
Was?	Lastenheft (L) Use Case (skizzenhaft; Was?)	
	Pflichtenheft (P) Use Case (detailliert; Wie?)	Product and sprint back log User stories (Was?) and tasks (Wie?)
Wer?	Domain expert (L/P) System analyst (L/P) Project manager (L/P) Development team (P)	Domain expert Product owner Scrum master Development team
Wann?	Analysephase	Sprint mit Analyse, Entwurf, Implementierung und Betrieb
Wie?	Erst Erstellung Pflichtenheft bevor Implementiert wird	Abfolge von 2- 4 Wochen Sprints
Wozu?	Planungssicherheit (Zeit, Kosten, Qualität relativ vollständig)	Kurze Feedbackzyklen um schnell auf Anforderungen reagieren zu können

# **Unified Modeling Language (UML) – Aktivitätsdiagramm (Activity diagram)**

## Aufgabe / Eigenschaften

- Ein Aktivitätsdiagramm bildet einen Anwendungsfall in Form seiner stattfindenden Aktivitäten und eventl. auch der dazu benötigten Objekte ab
- Der Ablauf der Aktivitäten wird mittels Kontrollstrukturen (Sequenz, Iteration, Verzweigung, Nebenläufigkeit) dargestellt
- Gehört zur dynamischen Sicht des Anwendungssystems

## Kontrollfluss

Der Ablauf von Aktivitäten eines Geschäftsprozesse (Use Case) kann durch einen sogenannten Kontrollfluss angegeben werden. Der Kontrollfluss stellt dar in welcher Reihenfolge die Aktivitäten ausgeführt werden. Zur Gliederung des Kontrollflusses werden sogenannte Kontrollstrukturen verwendet.

## Kontrollstrukturen

- Abfolge (Sequenz)
- Verzweigung (Alternative)
- Wiederholung (Schleife, Iteration)
- Nebenläufigkeit (Parallelisierung)

## Objektfluss

Nach dem sogenannten EVA-Prinzip (Eingabe, Verarbeitung, Ausgabe) sind nicht nur die Aktivitäten (Verarbeitung) von Bedeutung, sondern auch die Objekte, die für die Verarbeitung notwendig sind (Eingabe) und die von der Verarbeitung verändert oder erzeugt werden (Ausgabe). Wie die Objekte von einer zur nächsten Aktivität „fließen“ wird durch den Objektfluss dargestellt.

# Elemente Aktivitätsdiagramm

## Startknoten (start)

Der Beginn eines Anwendungsfalls (Use Case) wird durch den Startknoten angezeigt. Es kann mehrere Startknoten geben.



## Endknoten (end)

Das Ende eines Anwendungsfalls (Use Case) wird durch den Endknoten angezeigt. Es kann mehrere Endknoten geben.



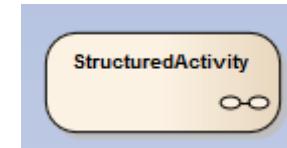
## Aktivität (activity)

Ein einzelner funktionaler Schritt eines Anwendungsfalls (Use Case) wird durch eine Aktivität dargestellt. Die Aktivität ist nicht mehr weiter verfeinerbar.



## Strukturierte Aktivität (structured activity)

Die Anordnung mehrerer Aktivitäten und deren Kontroll- und Objektfluss stellt eine Aktivität dar. Als Platzhalter für eine später näher zu beschreibende Aktivität dient die StructuredActivity.



## Kante

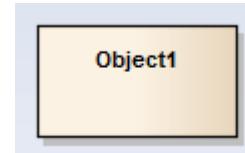
Die Abfolge von Aktivitäten wird durch einen Pfeil von einer Aktivität zur nächsten Aktivität dargestellt (Kontrollfluss). Um aufzuzeigen, welche Objekte als Input bzw. Output einer Aktivität vorhanden sind, wird ebenfalls ein Pfeil verwendet (Objektfluss)



# Elemente Aktivitätsdiagramm

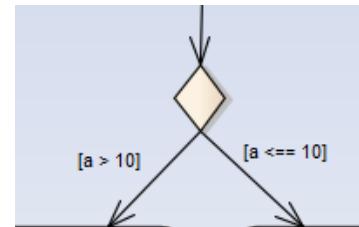
## Objekt

Objekte dienen als Input- oder Output-Parameter einer Aktion.



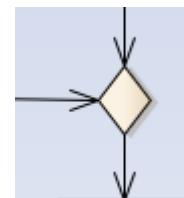
## Entscheidungsknoten (decision)

Die Abfolge von Aktivitäten kann an Bedingungen geknüpft sein. In Abhängigkeit von Bedingungen, sogenannten Guard-Conditions, welche in eckigen Klammern angegeben werden, wird die nachfolgende Aktivität bestimmt, an dessen eingehendem Pfeil die Bedingung gültig ist.



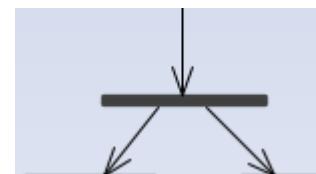
## Vereinigungsknoten

Die Zusammenführung vorher getrennten Kontrollflüsse geschieht durch einen Vereinigungsknoten. Zu jedem Entscheidungsknoten gibt es einen Vereinigungsknoten (Blockkonzept)



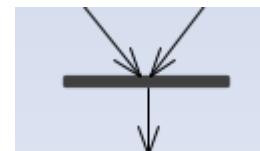
## Parallelisierungsknoten (split)

Aktivitäten können auch nebenläufig ablaufen. Ab dem schwarzen Balken ist der Kontrollfluss so aufgeteilt, dass in verschiedenen Strängen Aktivitäten parallel ablaufen. Es kann mehr als 2 Aktivitätsstränge geben.



## Synchronisationsknoten (merge)

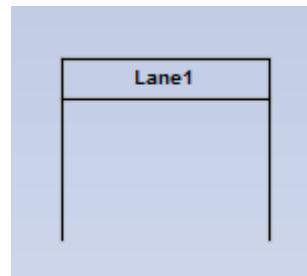
Die Zusammenführung nebenläufiger Aktivitätsstränge erfolgt durch den Synchronisationsknoten.



# Elemente Aktivitätsdiagramm

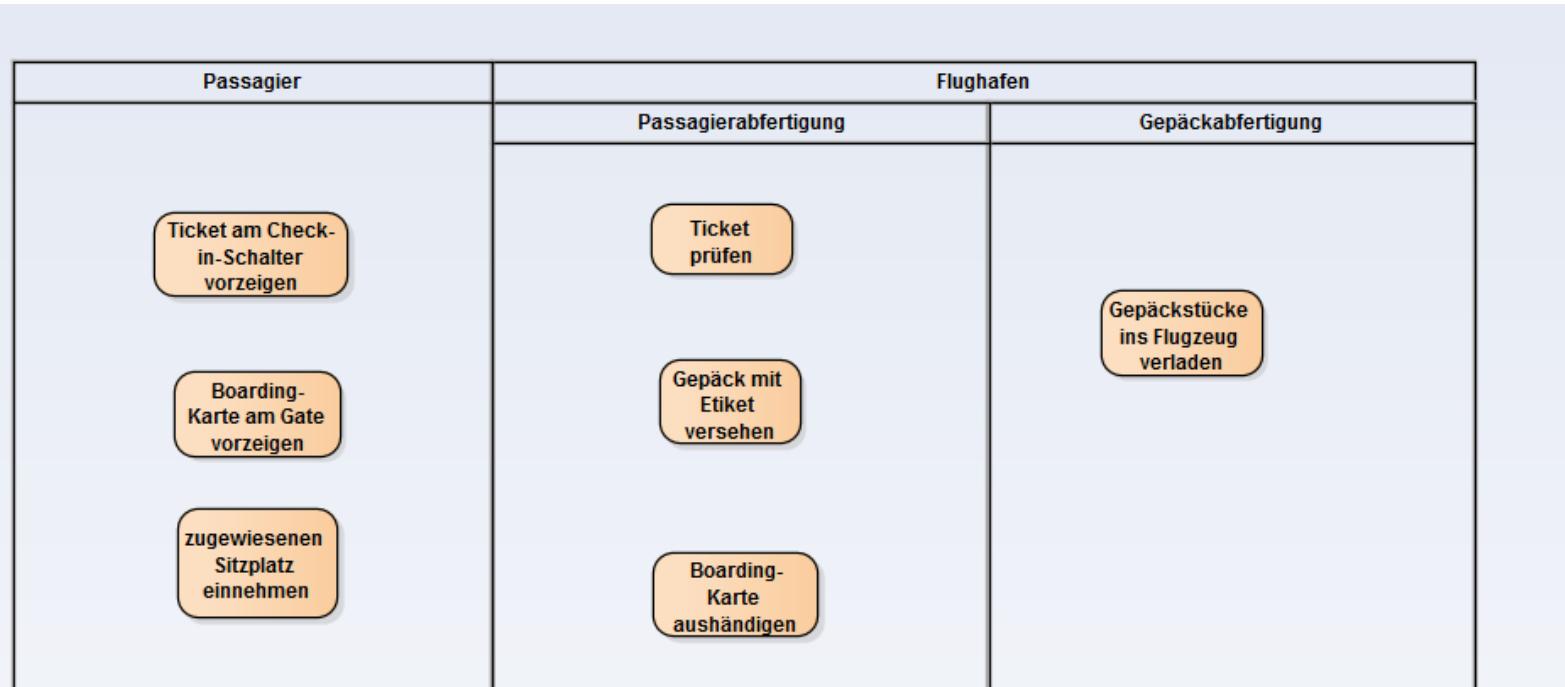
## Verantwortlichkeitsbereich (partition – „swimlane“)

Die Zuordnung von Aktivitäten zu einem Akteur oder zum betrachtenden System geschieht durch Verantwortungsbereiche. Der Name des Verantwortungsbereichs hat entweder den Namen des Akteurs oder des zu betrachtenden Systems. Verantwortlichkeitsbereiche können auch hierarchisch ineinandergeschachtelt werden.

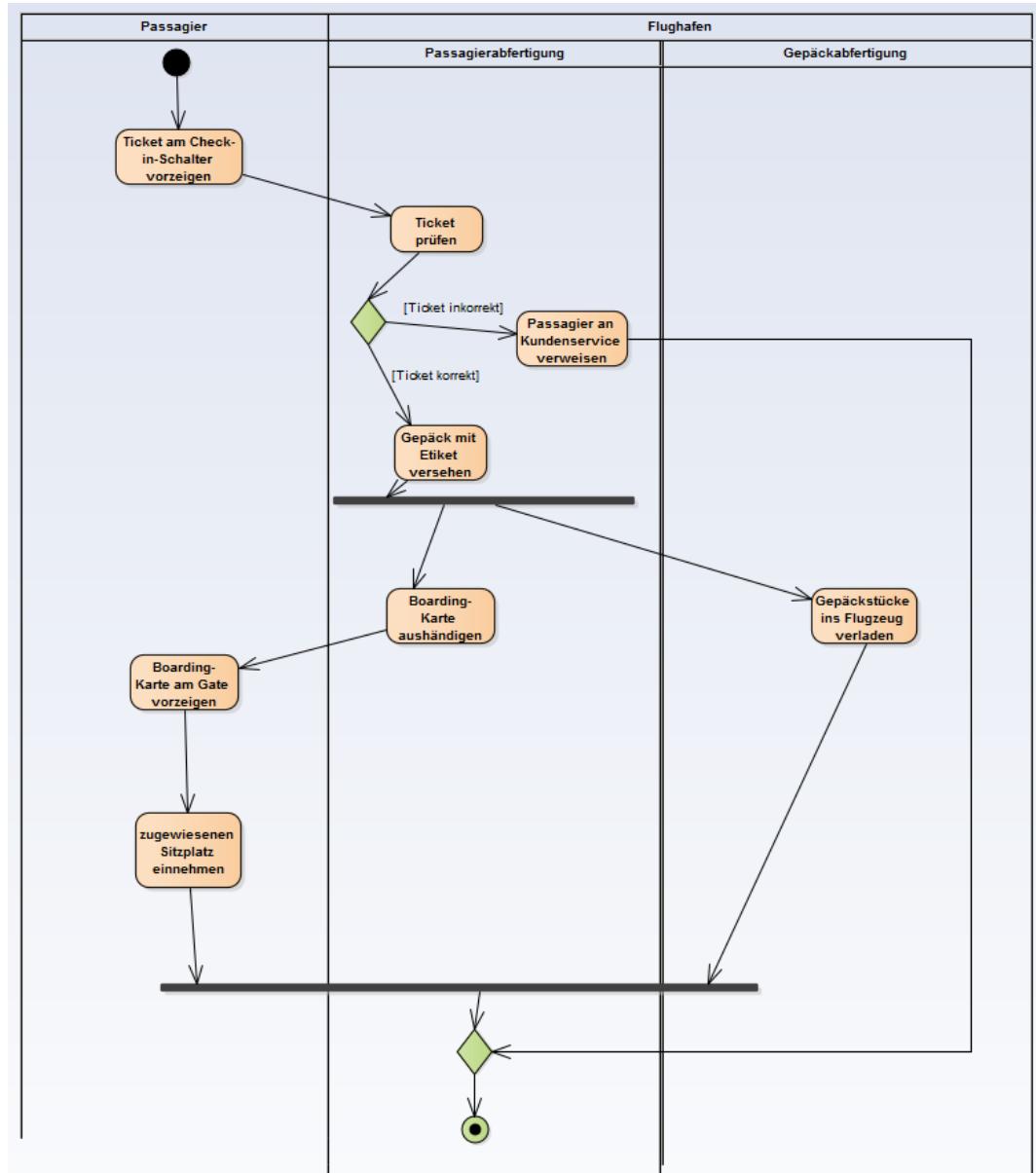


# Aktivitätsdiagramm - Beispiel

Use case „Check-in passenger“



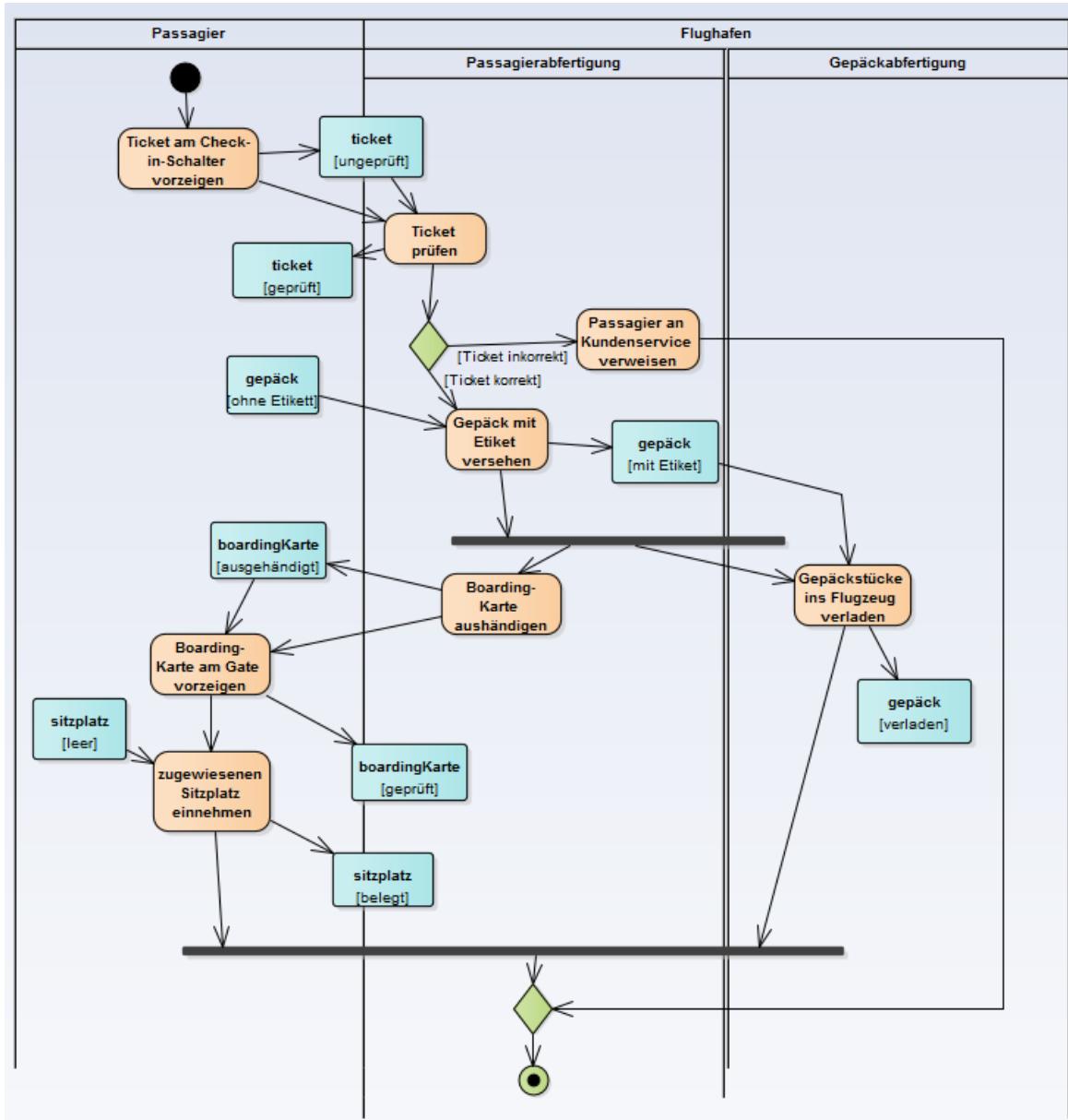
# Aktivitätsdiagramm - Beispiel



Use case „Check-in passenger“

Darstellung des Kontrollflusses

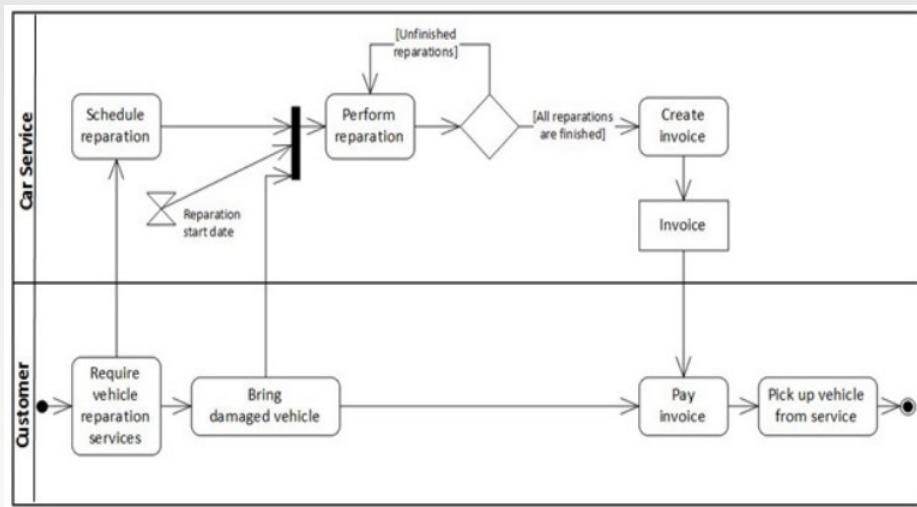
# Aktivitätsdiagramm - Beispiel



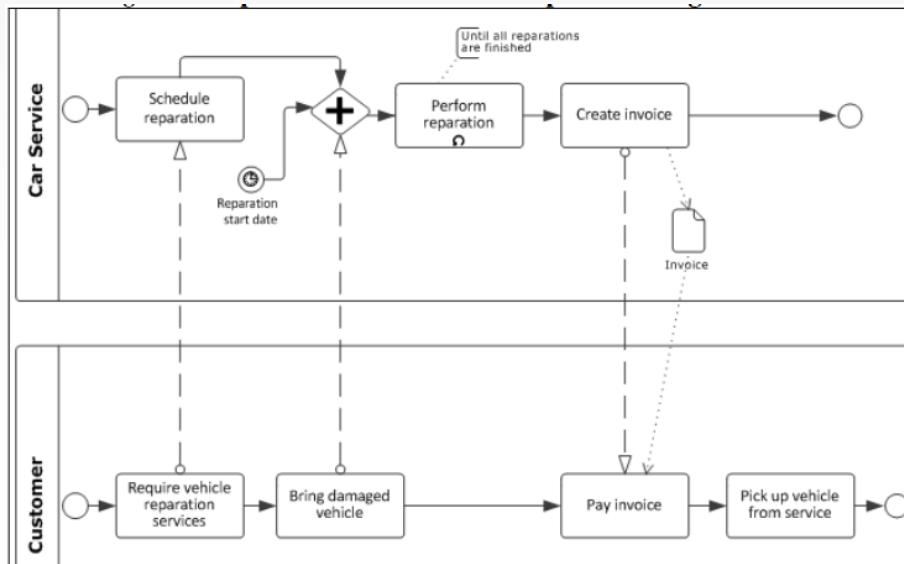
Use case „Check-in passenger“

Darstellung des Kontroll- und Objektflusses

UML  
activity diagram

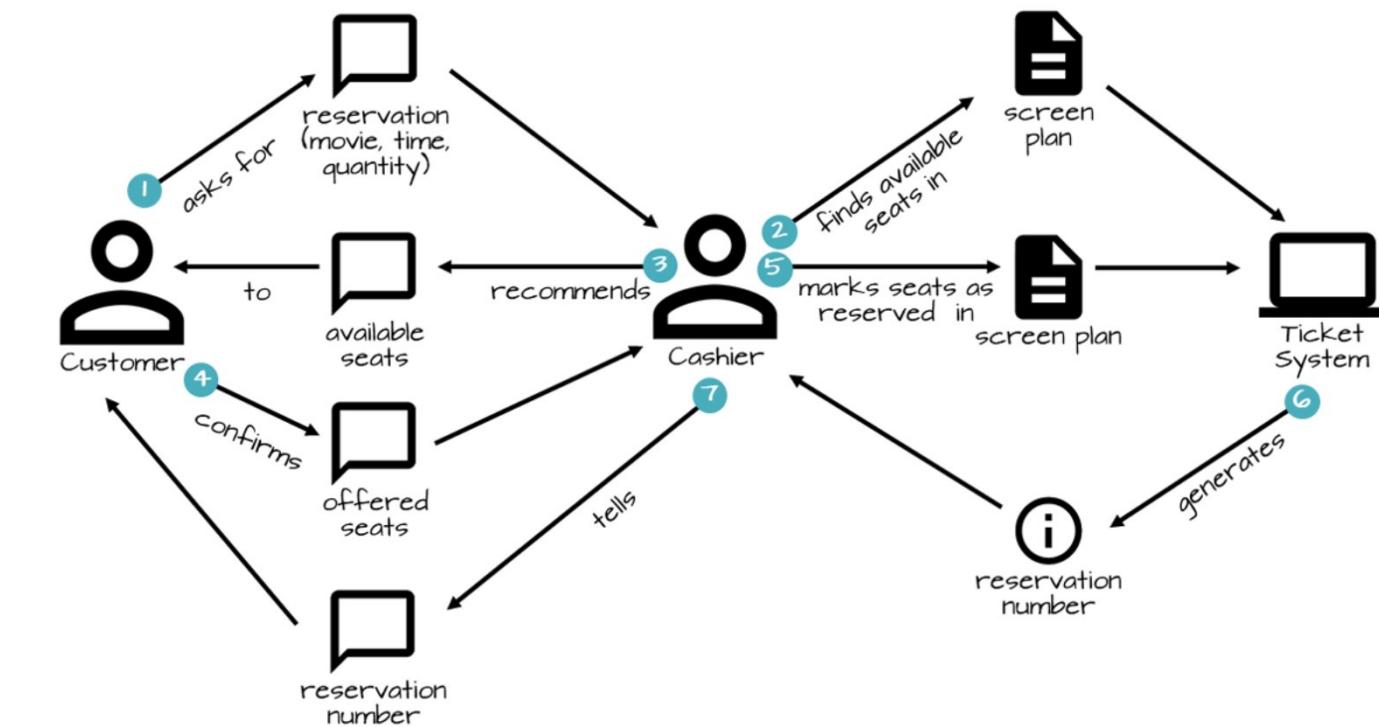


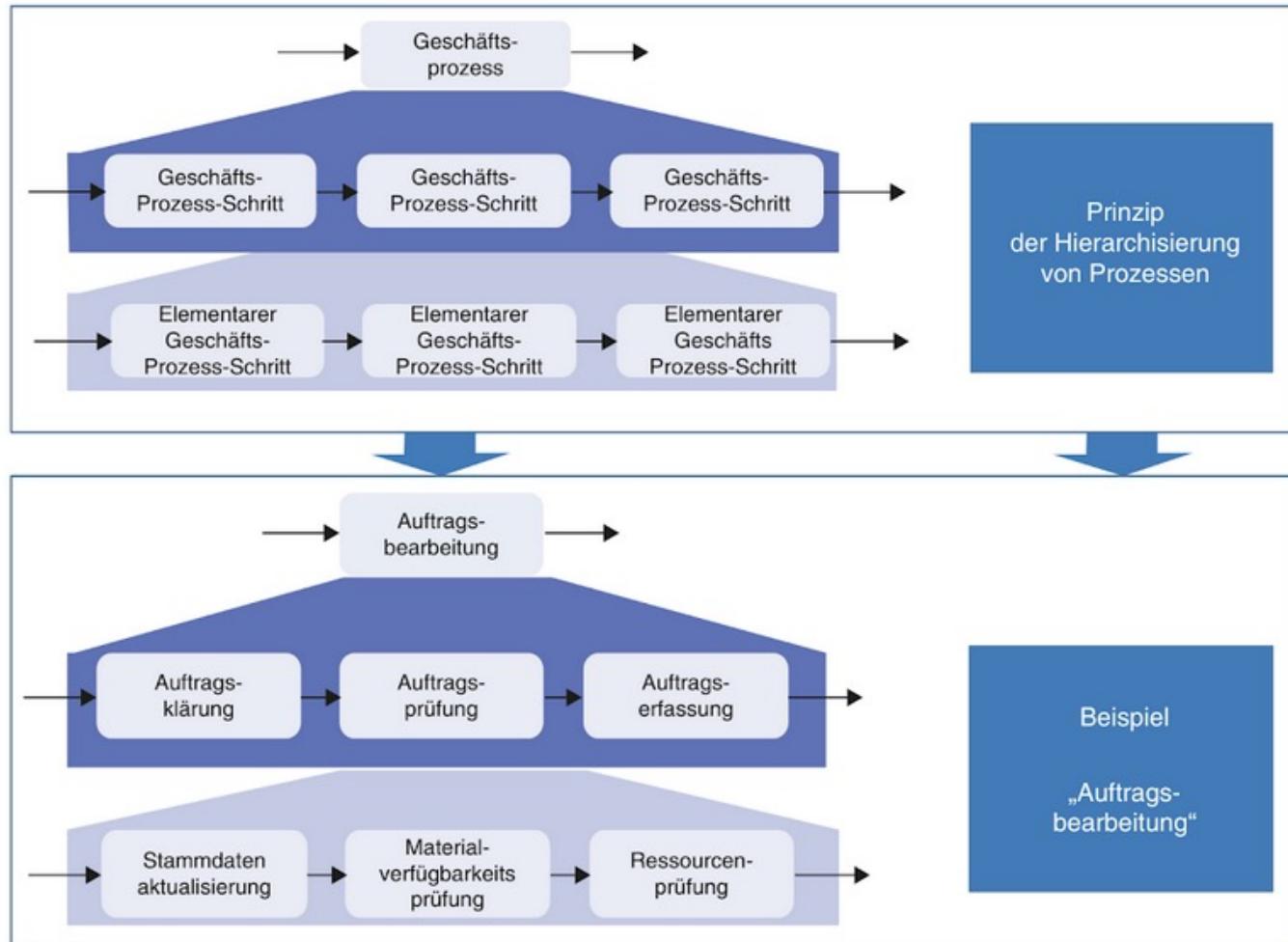
BPMN  
process diagram



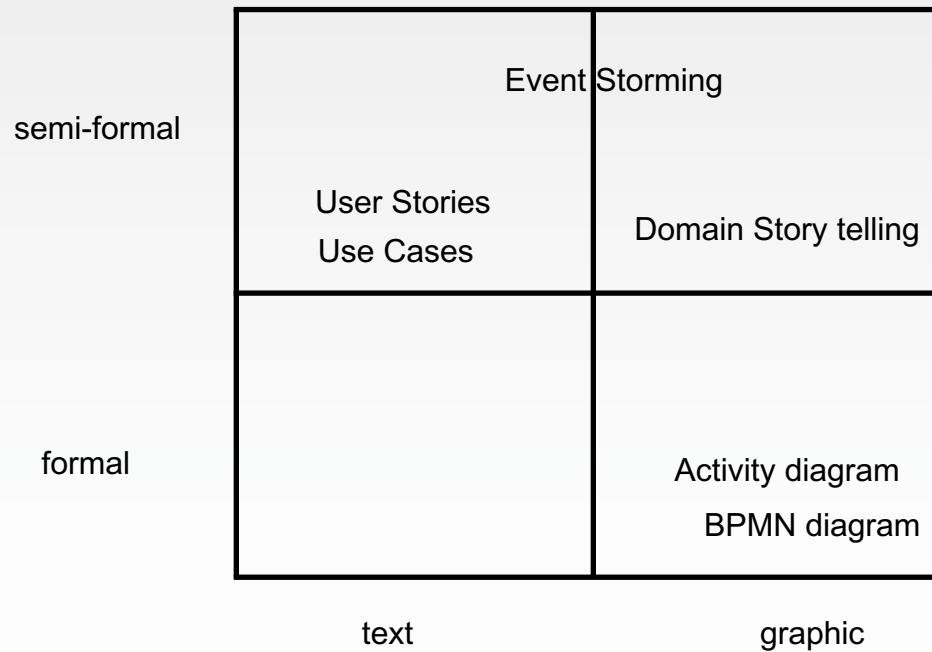
Quelle: Geambasu, C.; BPMN vs. UML activity diagram for business process modeling; 2012

# Domain story telling





Quelle: Gadatsch, A.; Grundkurs Geschäftsprozess-Management: Methoden und Werkzeuge für die IT-Praxis: Eine Einführung für Studenten und Praktiker, Vieweg, 8. Auflage, 2017

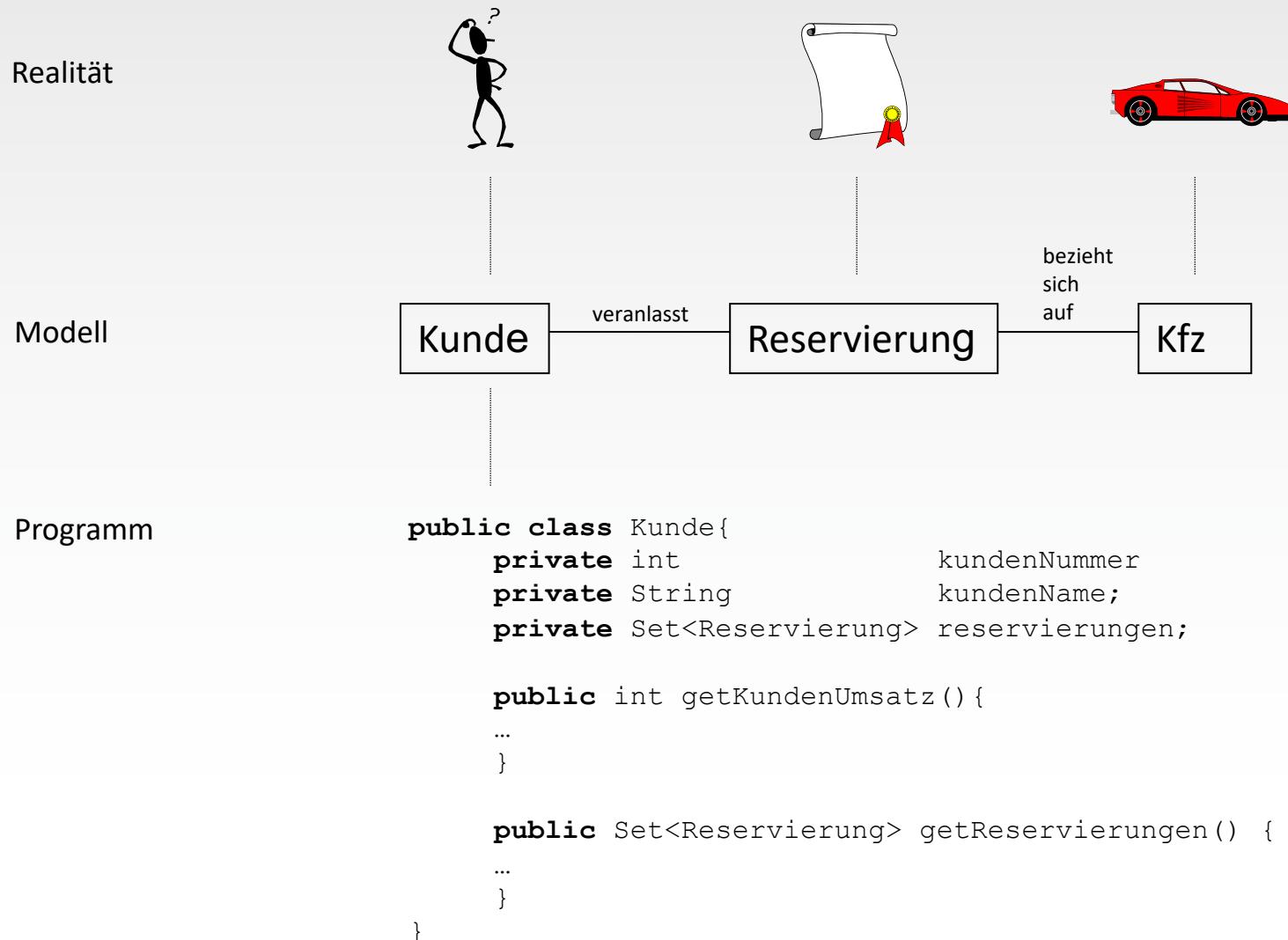


# **Unified Modeling Language (UML) – Klassendiagramm (Class diagram)**

## Aufgabe / Eigenschaften

- Die Fachbegriffe einer Anwendungsdomäne und ihre Beziehungen untereinander werden ohne ihre technischen Realisierungen dargestellt (Überblicksfachmodell)
- In einem späteren Schritt wird das Fachmodell um Attribute und Methoden erweitert (detailliertes Fachmodell)
- gehört zur strukturellen Sicht des Anwendungssystems

## Von der Realität zum Programm



## Objekt

- Jedes Objekt ist durch 3 Aspekte charakterisiert:
    - Identität
    - Verhalten
    - Zustand
- ➔ Die Identität eines Objektes wird durch einen eindeutigen Wert bestehende aus einem oder mehreren Attributwerten angegeben; 2 Objekte unterscheiden sich mindestens in ihrer Identität (häufig als ID definiert)
- ➔ Das Verhalten eines Objektes kann durch Nutzung von am Objekt aufrufbaren Methoden beeinflusst werden
- ➔ Der Zustand eines Objektes wird durch die Werte aller Attribute definiert

## Klasse

Eine Klasse besteht aus einer Sammlung von Attributen und Methoden, die den Zustand und das Verhalten ihrer *Instanzen (Objekte)* festlegen. Klassen sind durch Beziehungen mit anderen Klassen verbunden. Der Name einer Klasse muss innerhalb eines *Namensraumes* (z.B. Packet) eindeutig sein.

## Attribut

Eine gemeinsame Eigenschaft aller Objekte einer Klasse wird durch ein Attribut definiert. Ein Attributdefinition besteht aus *Name* und *Typ*.

## Methode

Eine gemeinsame Funktion aller Objekte einer Klasse wird durch eine Methode definiert. Eine Methode besteht aus einer *Signatur* (Angabe von Rückgabetyp, Methodename und Parameterliste) sowie aus einem *Methodenkörper*, der den Algorithmus beschreibt. *Abstrakte Methoden* haben keinen Rumpf.

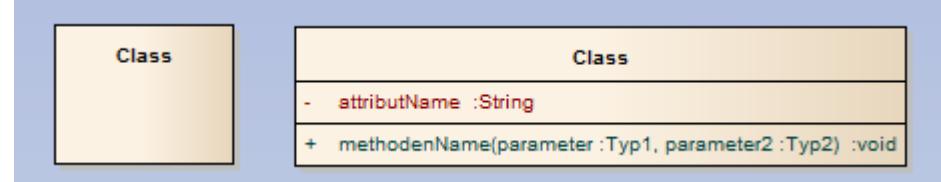
## Geheimnisprinzip, Kapselung

Attributwerte eines Objektes können nur durch Aufruf von Methoden gelesen oder geschrieben werden. Es soll keine direkte Manipulation der Werte unter Umgehung der Methoden möglich sein. Man spricht in diesem Fall vom Geheimnisprinzip bzw. Kapselung. Hierdurch wird es möglich die interne Darstellung (Wie?) von Attributwerten von seiner äußeren Darstellung (Was?) zu trennen, was erst die Austauschbarkeit von realisierten Datentypen ermöglicht.

# Elemente Klassendiagramm

## Klasse (class)

Ein reales oder abstraktes Konzept wird mittels eines Namens (Klassenname) ausgedrückt.  
Das Konzept hat Eigenschaften (Attribute) und Funktionen (Methoden) die es kennzeichnen.  
Zu einem Konzept kann es ein oder mehrere Ausprägungen (Objekte) geben.



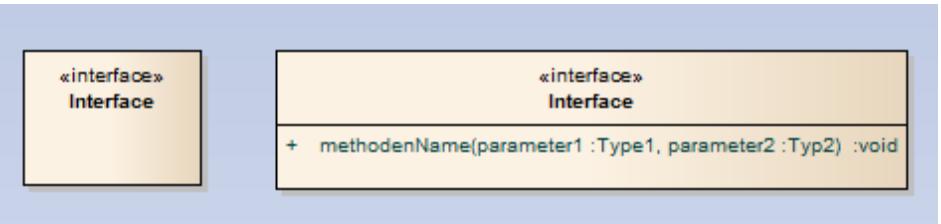
## Abstrakte Klasse (abstract class)

Soll von einer Klasse keine Instanz (Objekt) erzeugt werden aber dennoch in einer Generalisierung/Spezialisierungsstruktur verallgemeinert Attribute und Methoden angegeben werden, dann wird dies durch das Stereotyp <>abstract<> oder einem Klassennamen in kursiver Schrift angegeben.  
Eine abstrakte Klasse kann auch sowohl abstrakte wie konkrete (realisierte) Methoden enthalten.



## Schnittstelle (interface)

Eine Schnittstelle ist eine Sammlung von Methodensignaturen, also nicht implementierten Methoden. Ein Schnittstelle definiert „Was?“ ein Typ zu leisten hat. Klassen, die die Schnittstelle realisieren legen in unterschiedlicher Form fest „Wie?“ die Schnittstelle umgesetzt wird.



# Elemente Klassendiagramm

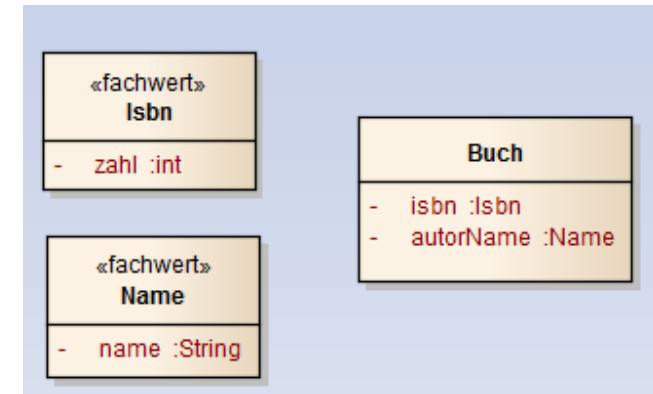
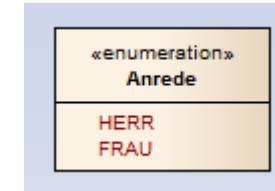
## Aufzählung/Enumeration (enumeration)

Ein Aufzählung oder Enumeration listet für einen Wertebereich die zulässigen Elementwerte auf. Enumerationswerte lassen sich nicht verändern und stellen damit Konstanten dar.

Der Stereotyp <>enumeration<> zeigt an, dass eine Aufzählung vorliegt. Die Werte der Enumeration werden durch Großbuchstaben angegeben.

## Fachwert (domain value)

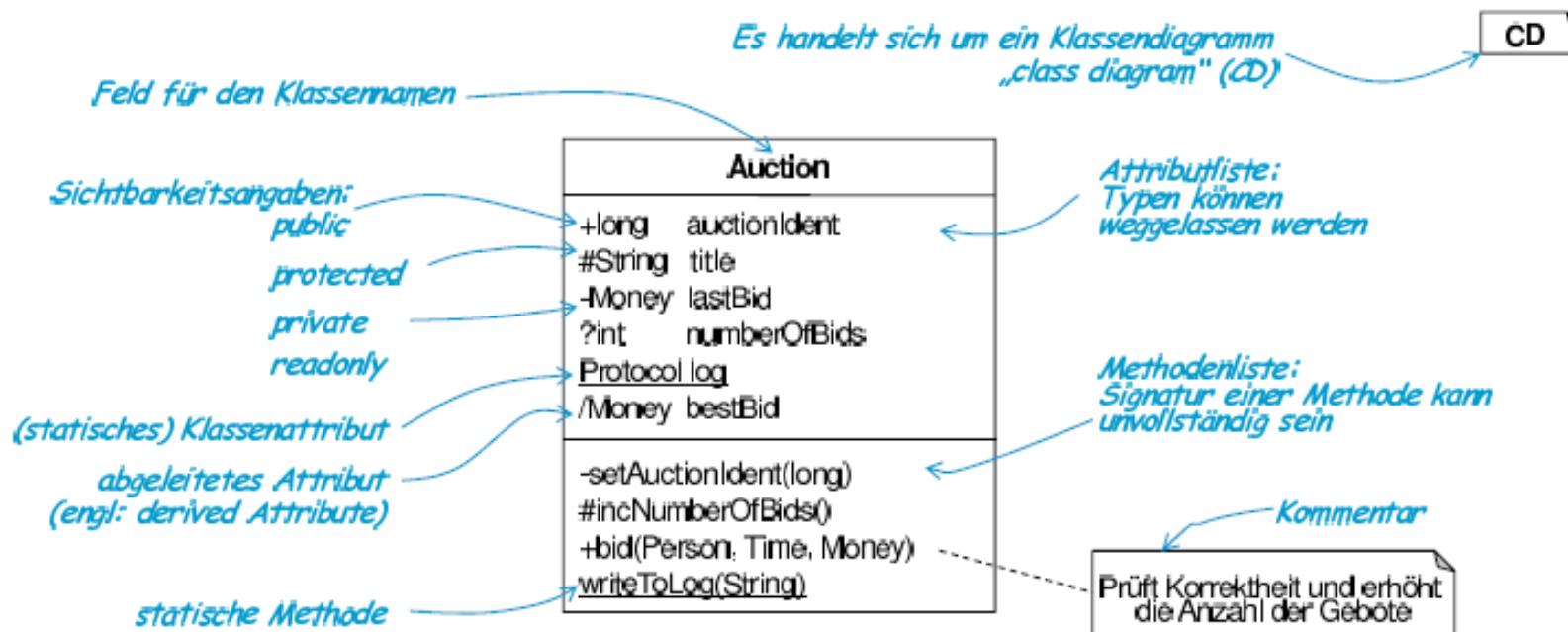
Ein Fachwert stellt das fachlich kleinste Element dar. Vom Modellierer erstellte (Fach-)klassen sollten als Typen von Attributten nur Fachwerte bzw. andere (Fach-)klassen verwenden. Durch Fachwerte ist der Übergang von fachlichen Namen und Typen zu technischen Namen und Typen realisiert. Hierdurch lässt sich durch des Geheimnisprinzips (Information hiding) an einer Stelle der technische Typ austauschen ohne , dass alle Klassen, die den Fachwert benutzen, dadurch geändert werden müssen.



# Elemente Klassendiagramm

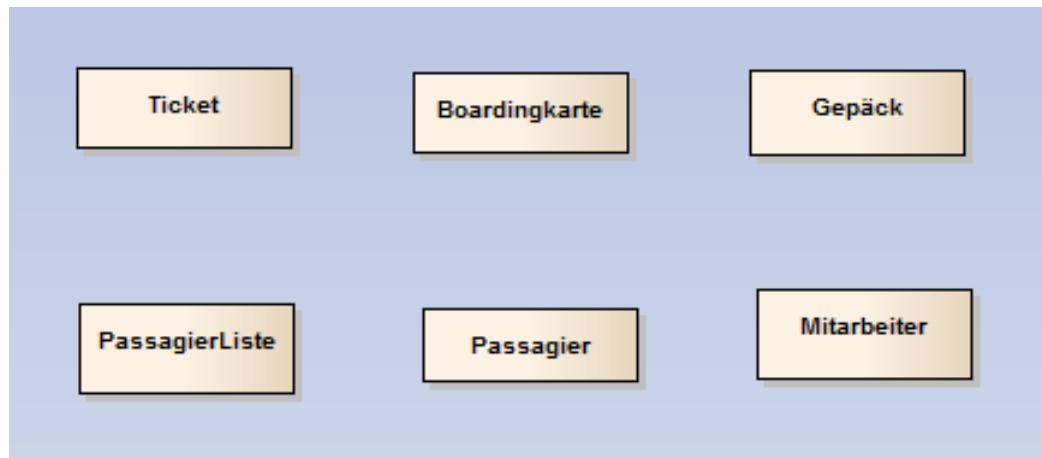
	<b>Klasse</b>	<b>abstrakte Klasse</b>	<b>Interface</b>
Instanzbildung	+	-	-
abstrakte Methoden	-	+	(+)
Instanzmethoden	+	+	-
Instanzvariablen	+	+	-
Konstanten	+	+	+
einfache Vererbung (extends)	+	(zwischen Klassen)	(zwischen Interfaces)
mehrfache Implementierung (implements)	+	(Klasse/Interface)	(Klasse/Interface)

# Klassendiagramm



# Klassendiagramm - Beispiel

## Fachklassen



## Klassendiagramme

Ein Klassendiagramm stellt eine Anordnung zwischen Elementen (Klassen, Schnittstellen, Enumerationen) und deren Beziehungen (Assoziation, Aggregation/Komposition) und Generalisierung/Spezialisierung) dar. Es gehört zu den Strukturdiagrammen in UML und bildet in Form eines Diagrammes ein Model des zu untersuchenden Systems ab.

## Fachklassendiagramme

Ein Fachklassendiagramm stellt die fachlichen Begriffe und deren Beziehungen in Form eines Klassendiagrammes dar. In dieser Unterart des Klassendiagrammes tauchen nur Beschreibungen (Klassen, Attribute, Methoden) aus dem zu untersuchenden fachlichen Bereich auf. Es gibt keine technische Begriffe. (Siehe Trennung von Zuständigkeiten (Separation of Concerns) bzw. Trennung von Was? (fachlich) und Wie? (technisch)). Es ist das initiale Klassendiagramm, welches in der Analysephase erstellt wird.

## Technische Klassendiagramme

In der Entwurfsphase wird das Klassendiagramm um technische Aspekte erweitert. Die fachlichen Sachverhalte aus dem Fachklassendiagramm bleiben aber vollständig erhalten.

# **Unified Modeling Language (UML) – Klassendiagramm - Beziehungsarten**

# Worauf es ankommt!

Was ist das?



Quelle: [http://www.siemund.org/blog/2012/01/20120120\\_04.jpg](http://www.siemund.org/blog/2012/01/20120120_04.jpg); Zugriff: 19.11.2012

# Worauf es ankommt!

Hätten Sie's gedacht?!



Quelle: [http://www.siemund.org/blog/2012/01/20120120\\_04.jpg](http://www.siemund.org/blog/2012/01/20120120_04.jpg); Zugriff: 19.11.2012

„Relations are entities which glue together other entities. Without relations the world would fall into many isolated pieces.“

Quelle: Guizzardi, G. et al.; Towards Ontological Foundations for UML Conceptual Models, 2002

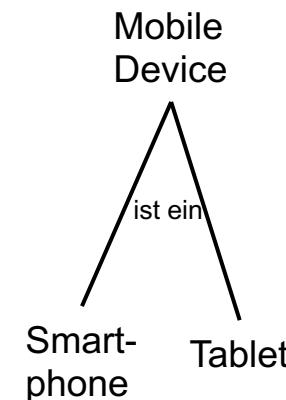
## Klassifizierung / Instanzierung



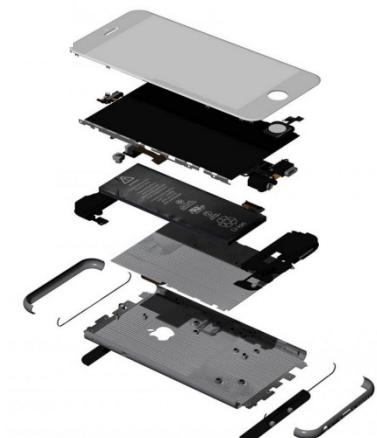
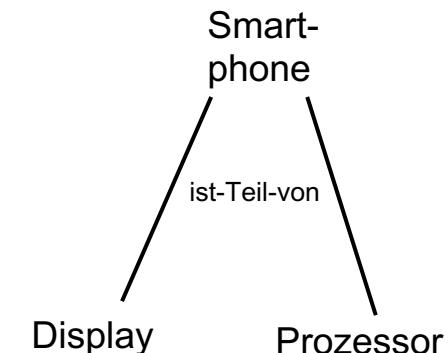
## Assoziation



## Generalisierung/ Spezialisierung



## Komposition/ Dekomposition



Quelle: [http://www.pocketpc.ch/magazin/wp-content/uploads/2014/08/iphone6\\_martinhajek\\_16.jpg](http://www.pocketpc.ch/magazin/wp-content/uploads/2014/08/iphone6_martinhajek_16.jpg);  
<http://www.tnooz.com/wp-content/uploads/2012/11/mobile-devices.jpg> ;  
<http://www.tabtech.de/wp-content/uploads/2011/11/galaxy-note-test-kopf.jpg>Zugriff: 22.09.2015

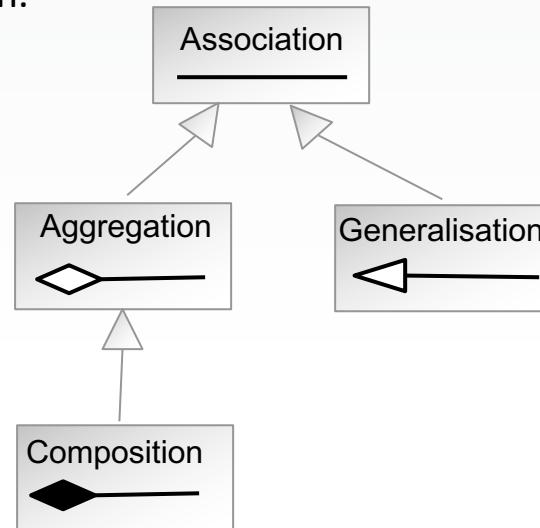
Bei der Festlegung von Beziehungen sind folgende Dinge festzulegen:

1. Beziehungsart
2. Beziehungsname (Leserichtung)
3. Mengenangabe
4. Gerichtetheit
5. Rolle

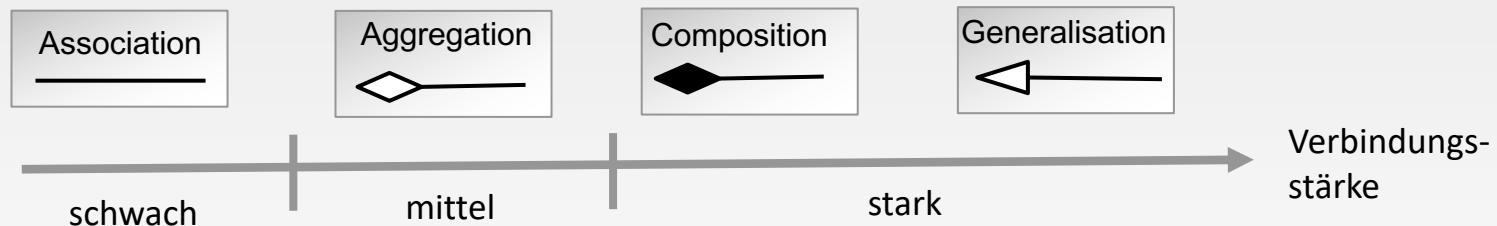
Für die Modellierung von Beziehungen zwischen Objekten bzw. Klassen sind 3 Beziehungsarten besonders wichtig:

- Assoziation
- Aggregation bzw. Komposition
- Generalisierung/Spezialisierung

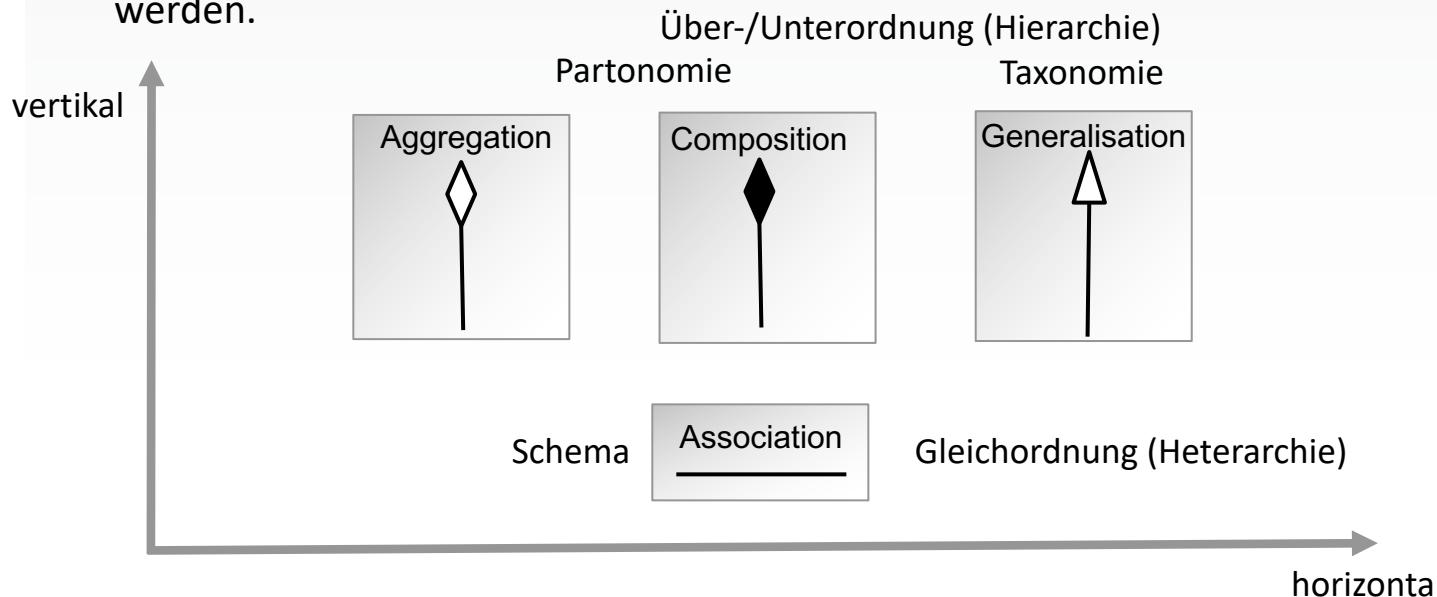
Während eine Assoziation bzw. eine Aggregation/Komposition bei einer Beziehung zwischen zwei Objekten vorkommt, handelt es sich bei der Generalisierung/Spezialisierung um eine Beziehung zwischen Klassen, die die Attribute und Methoden eines Objektes durch eine Generalisierung/Spezialisierung-Hierarchie (Taxonomie) bestimmen.



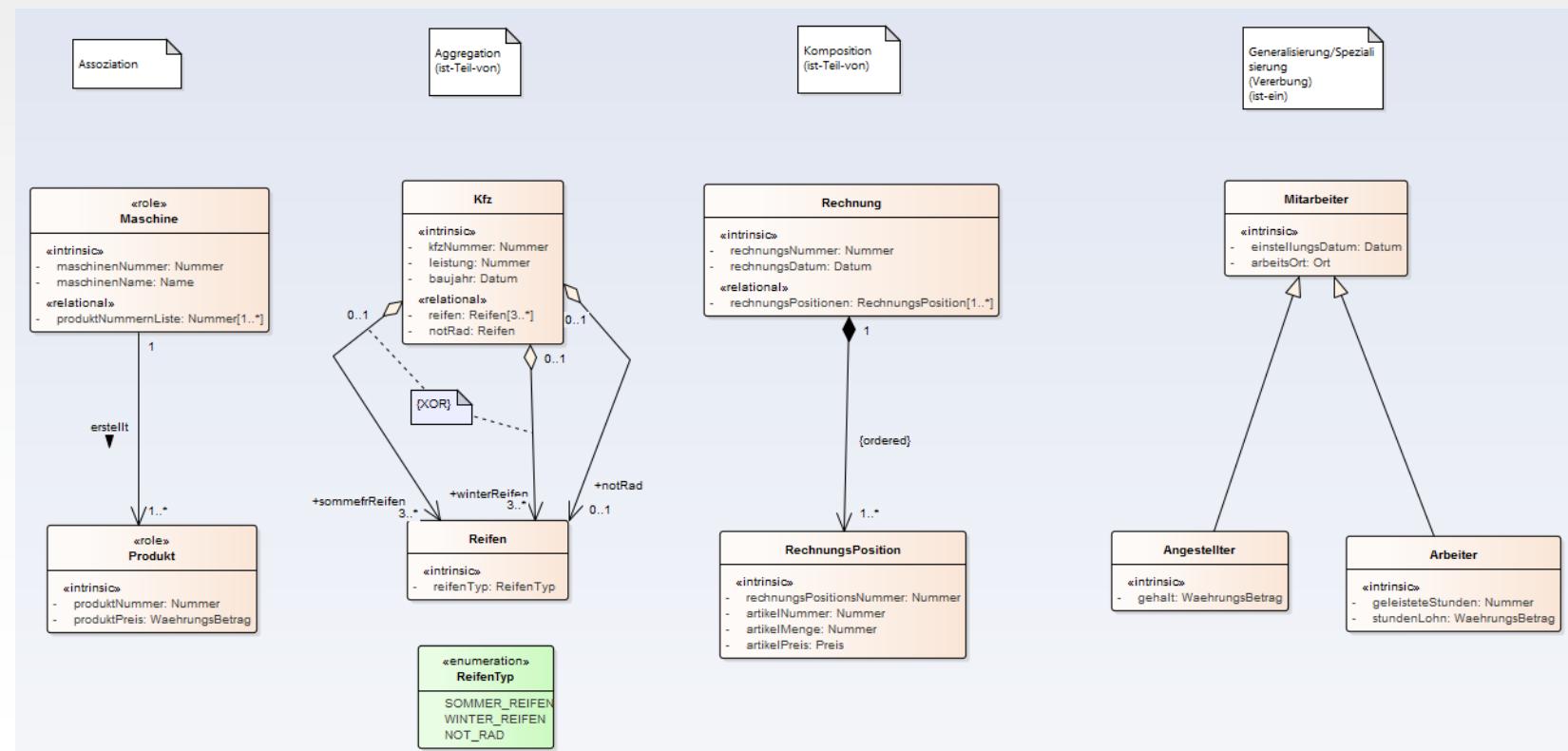
Die Beziehungsarten können folgendermaßen nach ihrer Stärke (ein Maß dafür wie lose die Verbindung ist) eingeordnet werden:



Die Beziehungsarten können nach vertikaler oder horizontaler Nutzung unterschieden werden.



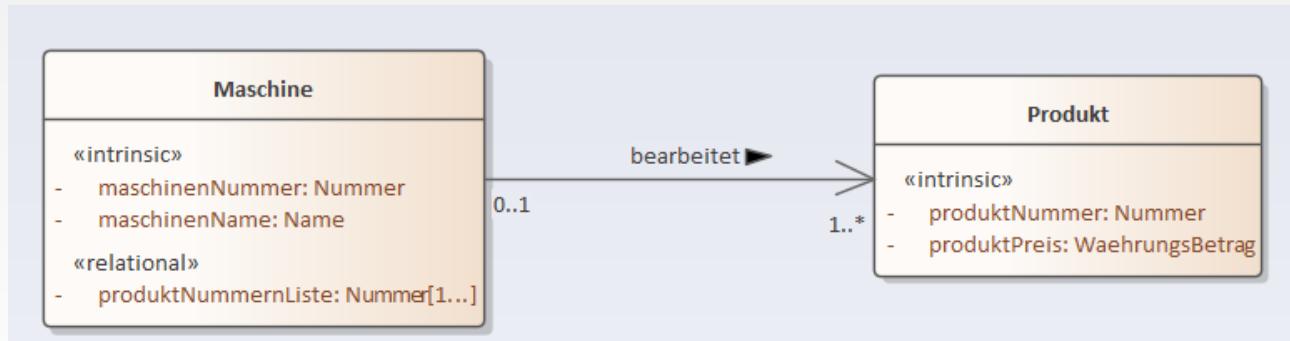
# Überblick Beziehungsarten



## Aufgabe

Die Assoziation ist die allgemeinste Beziehungsart. Wenn es überhaupt zwischen 2 Objekten eine Beziehung gibt, dann ist es eine Assoziation.

## Beispiel



## Elemente

Eine Assoziation beinhaltet eine VerbindungsLinie zwischen 2 Klassen (**Maschine** und **Produkt**), einen Assoziationsnamen (**bearbeitet**), eine Angabe zur Gerichtetheit (Pfeilspitze) sowie Mengenangaben (0..1 bzw. 1..\*).

## Regel



Zur Bestimmung von Mengenangaben gilt folgendes:

Immer von einer Seite mit „Ein“ anfangen wie z.B.:

Eine Maschine bearbeitet ein bis n Produkte.

Dann die Seite wechseln und wieder mit „Ein“ anfangen wie z.B.:

Ein Produkt wurde von keiner oder genau einer Maschine bearbeitet.

Mengenangaben gelten für Objekte obwohl sie im Klassendiagramm notiert werden!!!

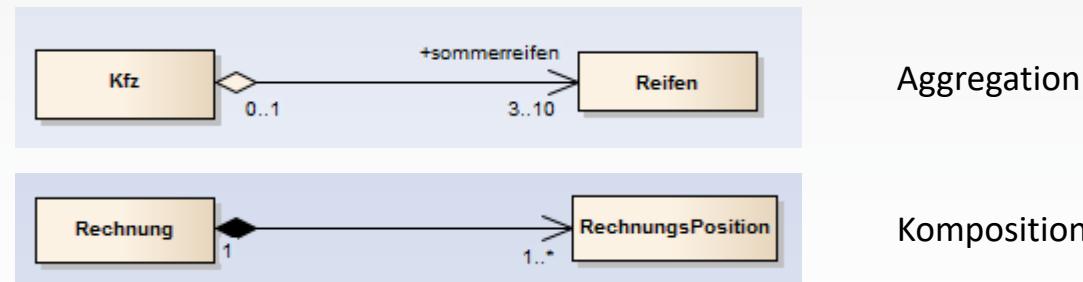
# Aggregation / Komposition (1)

## Aufgabe

Besteht zwischen Objekten eine Teil-Ganze-Beziehung (auch ist-Teil-von-Beziehung genannt), dann handelt es sich um eine Aggregation bzw. Komposition.

Der Komposition ist ein Spezialfall der Aggregation. D.h. sie unterscheidet sich von der Aggregation dahingehend, dass es noch ein zusätzliche Bedingung zwischen den Objekten, welche die Teile und dem Objekt, welches das Ganze darstellt, gibt. Können die „Teil-Objekte“ nicht ohne das „Ganze-Objekt“ existieren, dann handelt es sich um eine Komposition (Existenzabhängigkeit-Kriterium).

## Beispiel

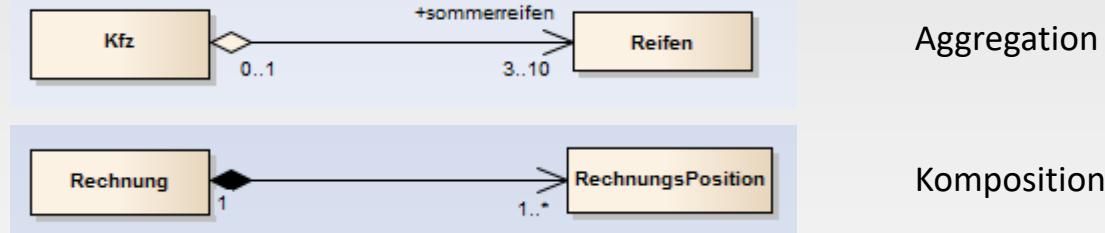


## Elemente

Die Raute auf der Seite des „Ganze-Objektes“ der dazugehörigen Klasse zeigt eine Aggregation bzw. Komposition an. Ist die Raute weiß, dann handelt es sich um eine Aggregation, ist die Raute schwarz, dann liegt eine Komposition vor.

Wie bei der Assoziation werden Mengenangaben (3..10 u.a.), Gerichtetheit (Pfeil) sowie Rollen (**sommerreifen**) notiert. Ein Beziehungsname ist überflüssig, da durch die Raute eindeutig eine ist-Teil-von-Beziehung angezeigt wird.

## Beispiel



## Erläuterung

### Vorbedingung

Es wurde erkannt das ein Objekt Teil eines anderen Objektes ist (ist-Teil-von-Beziehung)

### Argumentationsrichtung

Um zu prüfen, ob eine Beziehung zwischen 2 Objekten eine Aggregation oder eine Komposition ist, ist es entscheidend vom Teil auszugehen. Es wird gefragt, ob das Teil ohne dem Ganzen existieren kann. Ist das der Fall, handelt es sich um eine Aggregation ansonsten um eine Komposition.

### Existenzabhängigkeit

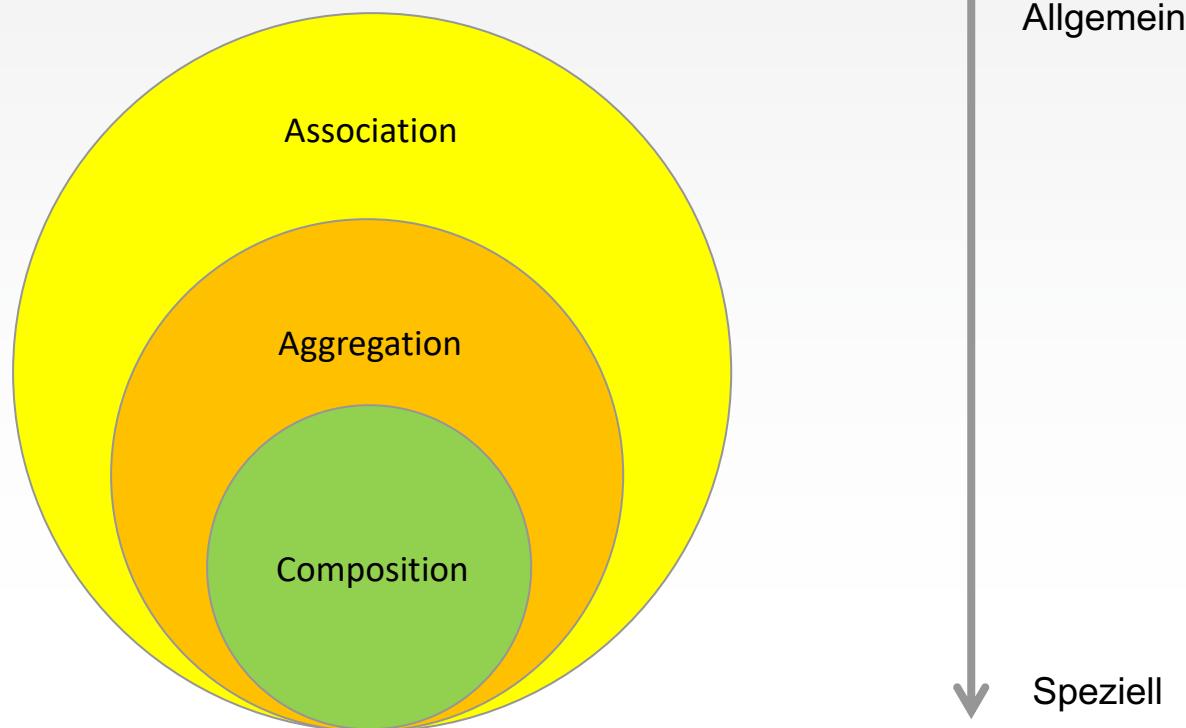
Die Existenzabhängigkeit des Teils wird durch die schwarze Raute dargestellt und der Multiplizität 1 bei dem Ganzen. Die Existenzabhängigkeit des Ganzen wird durch die Untergrenze 1 bei der Multiplizität des Teils dargestellt.

# Aggregation / Komposition (3)

„Aggregation conveys the thought that the whole is the sum of its parts. ...

Composition is a stronger form of aggregation that requires that a part be included in at most one whole at a time.“

Quelle: Olive, A.; Conceptual Modeling of Information Systems, Springer 2007, p. 142



Die Gerichtetheit (auch Direktonalität genannt) gibt an, ob von einem Objekt auf ein anderes zugegriffen werden kann. Mit Zugriff ist gemeint, dass die Methoden eines Objektes aufgerufen werden können.

Der Zugriff kann entweder direkt oder indirekt erfolgen.

Ein direkter Zugriff liegt vor, wenn ein Objekt der Klasse A ein Attribut vom Typ der Klasse B hat (Aggregation) und daher direkt an dem in der Klasse A befindlichen Objekt vom Typ B Methoden aufgerufen werden können.

Ein indirekter Zugriff liegt vor, wenn ein Objekt der Klasse A ein Attribut hat, dass den eindeutigen Schlüssel (Id) der Klasse B darstellt (fachliche Referenz, Assoziation). Hier kann ein Objekt der Klasse A nur durch Suche aufgrund des Schlüssels ein Objekt vom Typ B aus der Menge aller Objekte der Klasse B ermitteln. Erst danach kann eine Methode am Objekt vom Typ B aufgerufen werden.

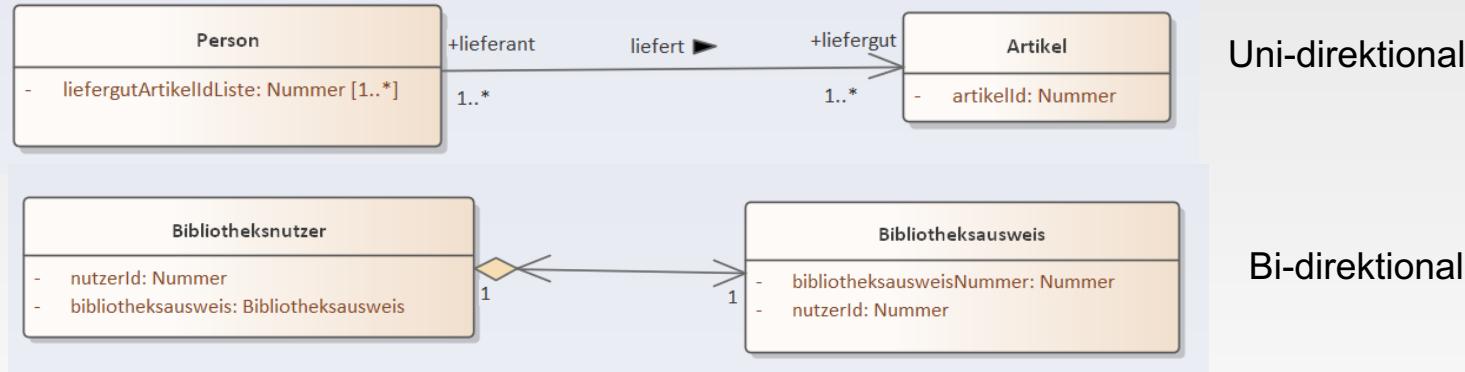
Man unterscheidet zwischen

- Uni-direktonal
- Bi-direktonal

Uni-Direktonalität liegt vor, wenn Objekte der Klasse A auf Objekte der Klasse B zugreifen können, aber nicht umgekehrt.

Bi-Direktonalität liegt vor, wenn sowohl Objekte der Klasse A auf Objekte der Klasse B und umgekehrt zugreifen können.

## Beispiel



Gibt es keine Pfeile an einer Assoziation bzw. Aggregation/Komposition, kann davon ausgegangen werden, dass es sich um eine bi-direktionale Beziehung handelt.

Um festzulegen mit wie vielen Objekten ein Objekt in Beziehung tritt, ist es notwendig Mengenangaben festzulegen. Die Definition der Mengenangaben wird auch Multiplizität oder Kardinalität genannt.

Die Definition von Mengenangaben kommt nur bei Assoziation und Aggregation/Komposition vor. Nicht bei Generalisierung/Spezialisierung !!!

Bei der Festlegung von Mengen wird sowohl eine Unter- wie eine Obergrenze definiert.

1	<input type="text"/>	genau 1
0..1	<input type="text"/>	0 bis 1
*	<input type="text"/>	0 bis viele
3..*	<input type="text"/>	3 bis viele
0..2	<input type="text"/>	0 bis 2
2	<input type="text"/>	genau 2
2, 4, 6	<input type="text"/>	2, 4 oder 6
1..5, 8, 10..*	<input type="text"/>	nicht 6, 7 oder 9

# Multiplizität (Kardinalität) (2)

## Beispiel (mit Attributen)



Anzahl 1



Anzahl n

## Aufgabe

Eine Rolle definiert Attribute und Methoden unter einem Namen, die ein Objekt innerhalb einer Beziehung hat.

## Beispiel



## Elemente

Eine Rolle kann in UML entweder an einem Assoziations- bzw. Aggregations-/Kompositionsende angegeben werden (z.B. `diensttelefon`, `mitarbeiter`) oder als selbstständige Klasse (siehe Role-Object-Pattern) definiert werden.

„a role is a mask that an object can wear“

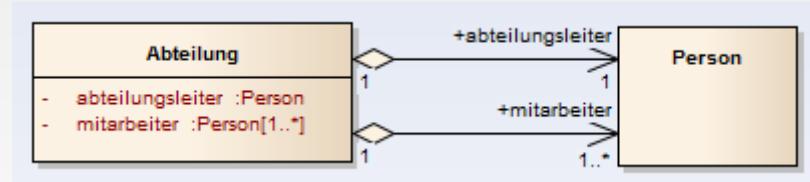
F. Steimann

## Roles

1. A role comes with its own properties and behaviour
2. Roles depend on relationships
3. An object may play different roles simultaneously
4. An object may play the same role several times, simultaneously
5. An object may acquire and abandon roles dynamically
6. The sequence in which roles may be acquired and relinquished can be subject to restrictions
7. Objects of unrelated types can play the same role
8. Roles can play roles
9. A role can be transferred from one object to another
10. The state of an object can be role-specific
11. Features of an object can be role-specific
12. Roles restrict access
13. Different roles may share structure and behaviour
14. An object and its roles share identity
15. An object and its roles have different identities

Zu einer Klasse kann es mehrere Assoziationen bzw. Aggregationen/Kompositionen geben.  
Dies ist vor allem dann sinnvoll um Rollen zu unterscheiden.

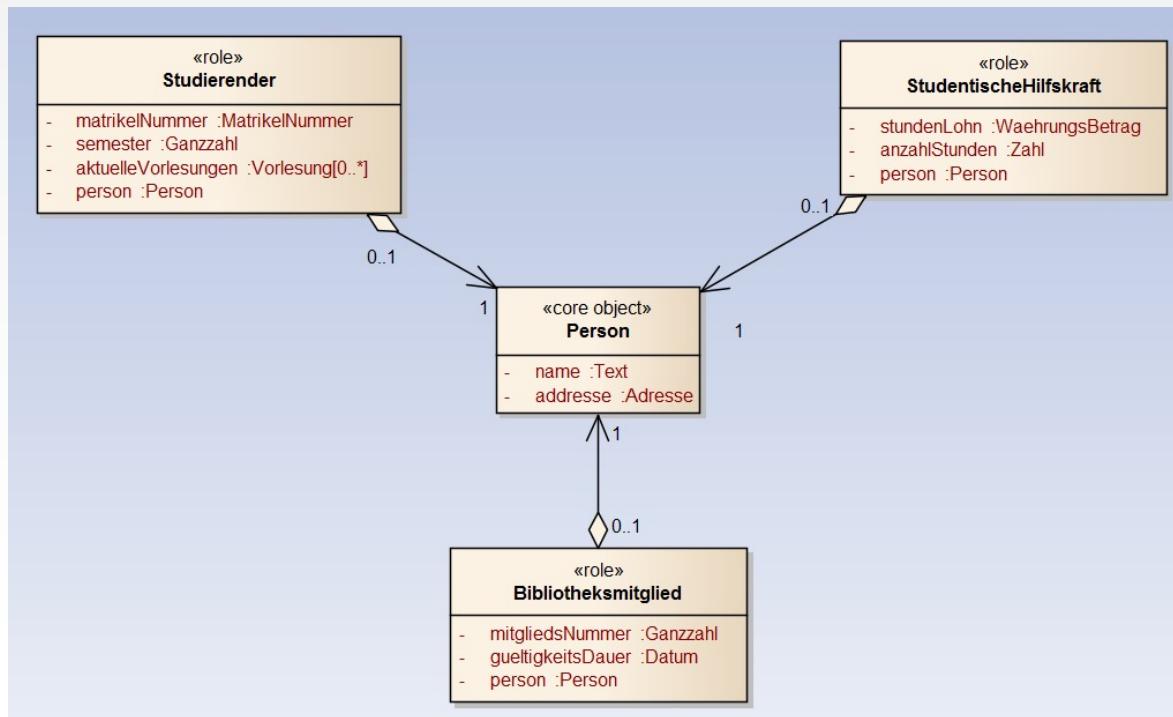
## Beispiel



## Role-Object Pattern

Das Role-Object Muster gehört zu den sogenannten „Analysemuster“, d.h. einer immer wieder vorkommenden Anordnung von Klassen in unterschiedlichen Fachmodellen. Es gibt mehrere Möglichkeiten das Role-Object-Pattern zu modellieren und zu implementieren.\*

### Beispiel



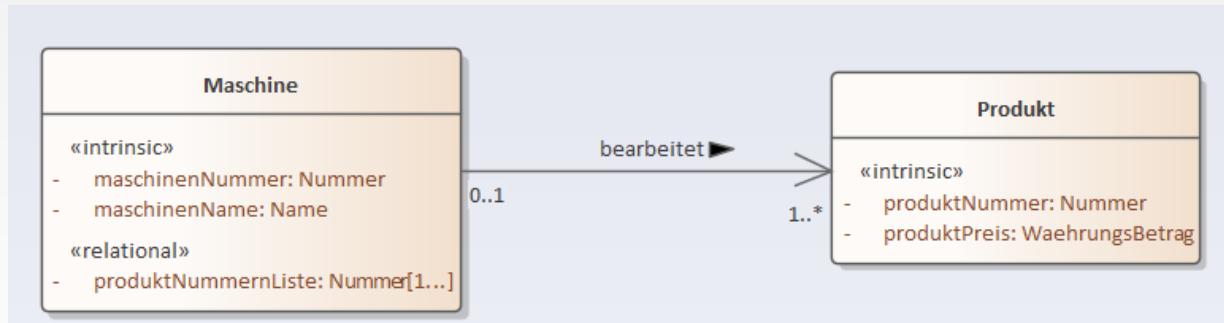
\* Siehe Bäumer, D. et al; The Role Object Pattern, PLoP 97; <http://st-www.cs.illinois.edu/users/hanmer/PLoP-97/Proceedings/riehle.pdf>

04.07.24     Fowler, M.; Dealing with Roles; <http://www.martinfowler.com/apsupp/roles.pdf>

Attribute einer Klasse, die unabhängig von Beziehungen vorhanden sind, heißen „intrinsic attributes“.

Eine Beziehung zwischen Objekten zweier Klassen wird durch ein sogenanntes „relational attribute“ dargestellt.

## Beispiel



Die Attribute `maschinenNummer` und `maschinenName` der Klasse **Maschine** sind „intrinsic attributes“, da Objekte dieser Klasse durch diese Attribute unabhängig von Beziehungen beschrieben werden.

Das Attribut `produktNummernListe` der Klasse **Maschine** ist ein „relational attribute“, da dieses Attribut die Beziehung zwischen Objekten der Klasse **Maschine** und Objekten der Klasse **Produkt** wiedergibt.

Es handelt sich um eine uni-direktionale Beziehung von der Klasse **Maschine** zur Klasse **Produkt**, da es nur „relational attributes“ bei der Klasse **Maschine** nicht aber bei der Klasse **Produkt** gibt.

“... by intrinsic properties I mean the properties that each entity has of itself, despite its extrinsic relations with other entities, and relational properties are properties that each entity has and acquires due to its extrinsic relations with other entities in its environment.”

Quelle: Santos, G.; Upward and Downward Causation from a Relational-Horizontal Ontological Perspective, 2015

„An intrinsic property is a property that a thing has of itself ... An extrinsic (or relational) property is a property that depends on a thing's relationship with other things.“

Quelle: [https://en.wikipedia.org/wiki/Intrinsic\\_and\\_extrinsic\\_properties\\_\(philosophy\)](https://en.wikipedia.org/wiki/Intrinsic_and_extrinsic_properties_(philosophy)); 01.07.2023

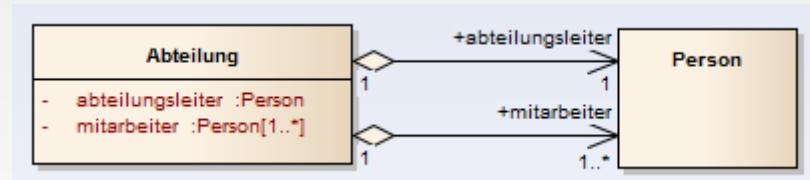
„a property of an object-system is considered as being intrinsic (and hence non-relational) if the object has this property in and of itself, independently of the existence of other objects“

Quelle: Karakostas, V., Atomism vs holism in science and philosophy, 2008

Die Beziehung zwischen Objekten kann direkter (Aggregation / Komposition) bzw. indirekter (Assoziation) Art sein.

## Beispiel

Direkt



Indirekt



Bei der direkten Beziehung enthält ein Objekt ein anderes. So besteht eine Abteilung aus einem Abteilungsleiter und ein oder mehreren Mitarbeiter.

Bei der indirekten Beziehung enthält ein Objekt nur eine fachliche Referenz (einen fachlichen Schlüssel), der auf ein Attribut des referenzierten Objektes verweist. So referenziert ein Auftrag den Kunden über seine Kundennummer, d.h. der Auftrag enthält nicht das gesamte Kunden-Objekt, sondern aufgrund der im Auftrag vermerkten Kundennummer nur einen Verweis auf den Kunden.

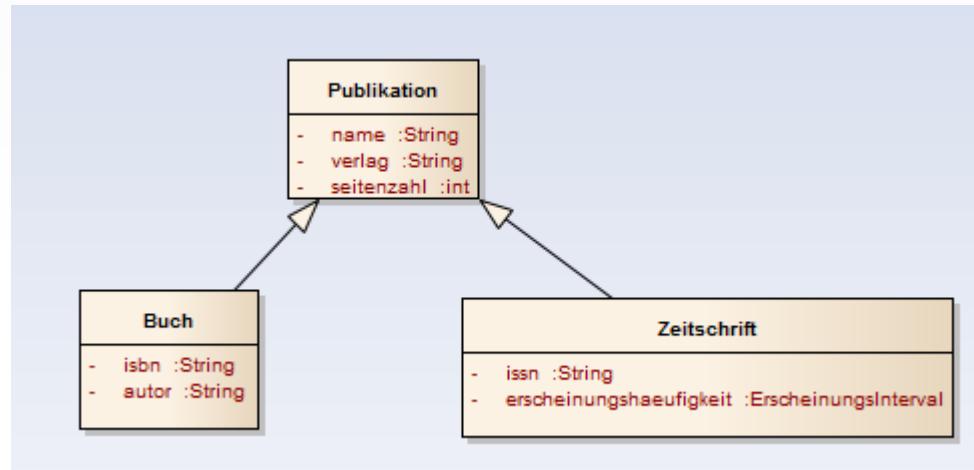
## Aufgabe

Besteht zwischen Klassen eine Beziehung in der Form, dass Attribute und Methoden von einer Oberklasse an eine Unterklasse weitergegeben werden und in der Unterklasse weitere dazukommen, dann handelt es sich um eine Generalisierung/Spezialisierung. Aus Sicht der Unterklassen werden gemeinsame Attribute und Methoden in der Oberklasse generalisiert (Abstraktionsprinzip), aus Sicht der Oberklasse wird durch Hinzufügen von Attributen und Methoden Unterklassen spezialisiert.

Zwischen Unterklasse und Oberklasse besteht eine „ist-ein(e-Art-von)-Beziehung“.

Eine Generalisierung/Spezialisierung ist nur zwischen Klassen und nicht zwischen Objekten möglich !!!

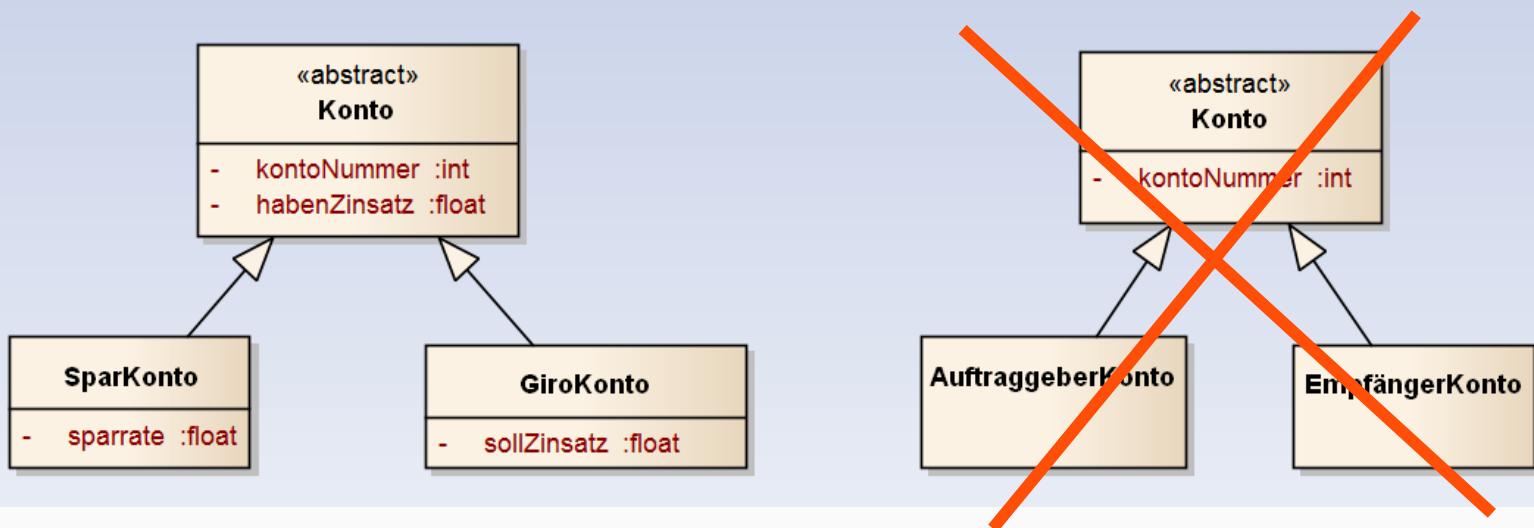
## Beispiel



„Generalization and specialization are two different views of the *IsA* relationship, one from the supertype and the other from the subtypes.“

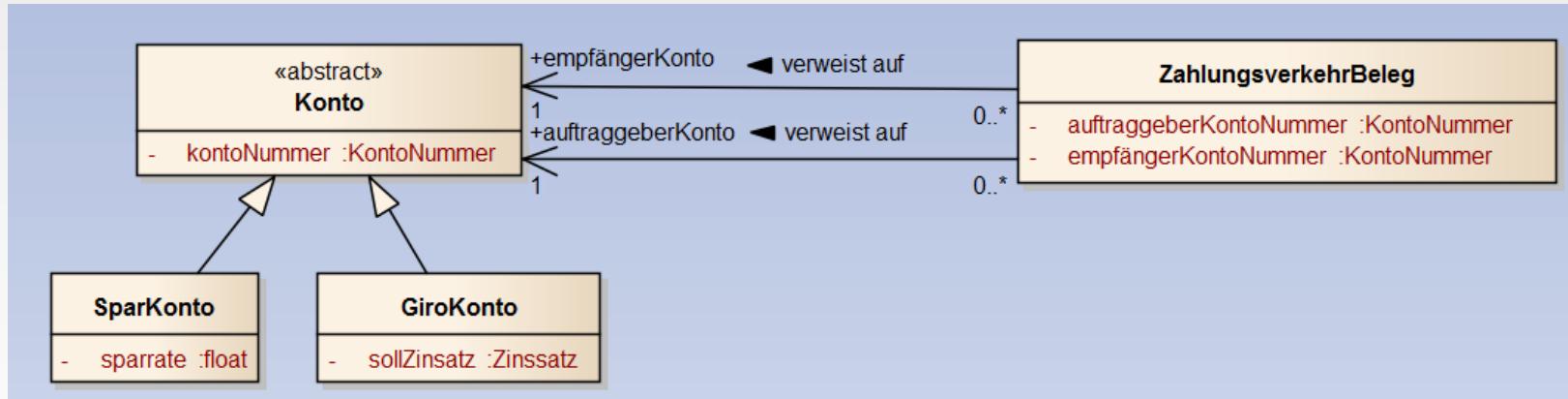
Quelle: Olive, A., Conceptual Modeling of Information Systems, 2007 p, 217

# Generalisierung/Spezialisierung vs. Rolle



Im Gegensatz zu SparKonto und GiroKonto sind AuftraggeberKonto und EmpfängerKonto keine Unterklassen von Konto. AuftraggeberKonto und EmpfängerKonto sind Rollen eines Kontos, da ja ein Konto sowohl AuftraggeberKonto wie EmpfängerKonto sein kann.

Man erkennt an den fehlenden Attributen bei AuftraggeberKonto und EmpfängerKonto in der ursprünglich anvisierten Gen-Spec-Hierarchie, dass es sich um Rollen handeln muss.



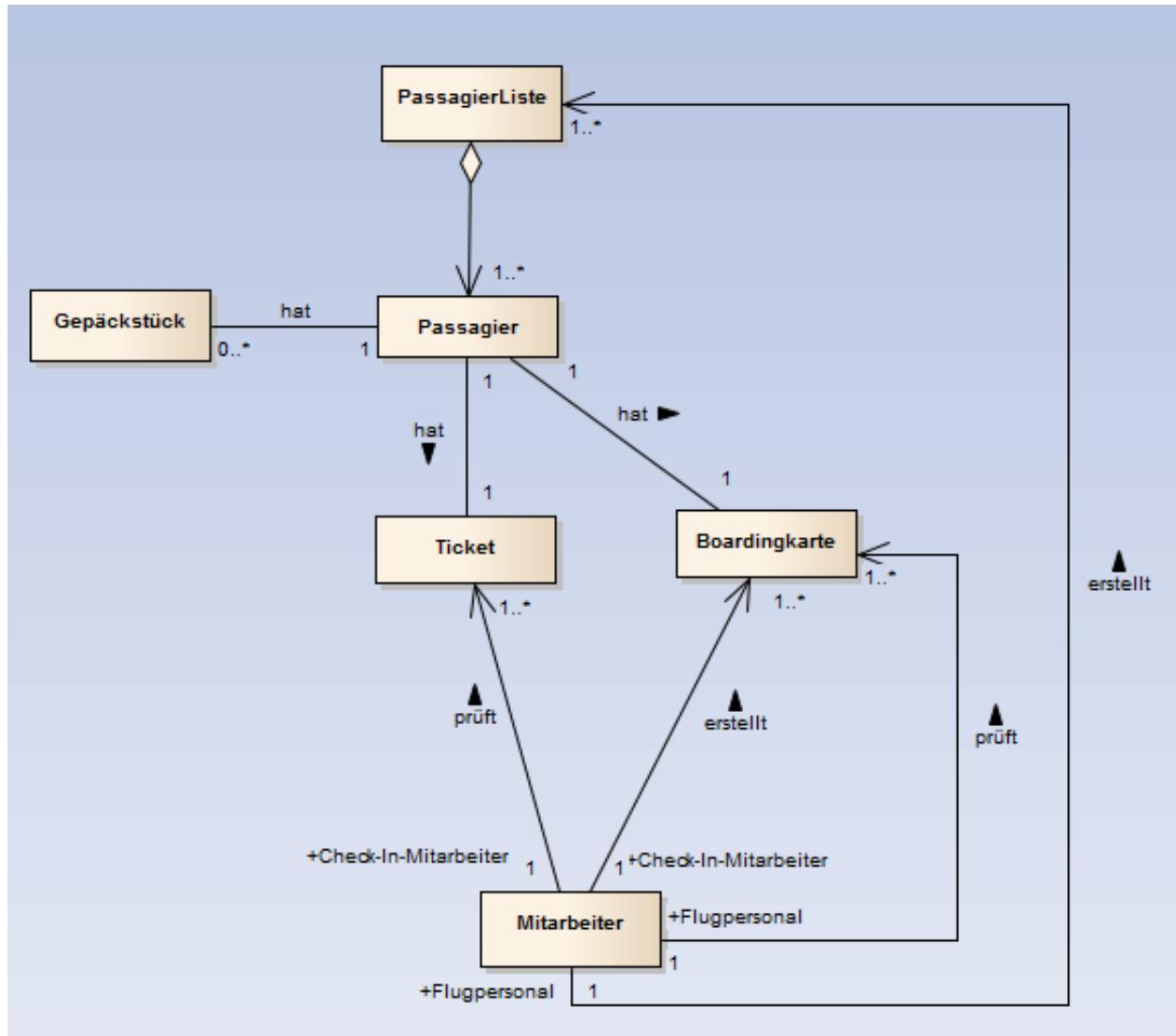
In der Klasse **ZahlungsverkehrBeleg** wird durch die Attribute **auftraggeberKontoNummer** bzw. **empfaengerKontoNummer** auf das jeweilige Konto mit entsprechender **kontoNummer** verwiesen. Es handelt sich um Assoziation zwischen **Konto** und **ZahlungsverkehrBeleg**, da in **ZahlungsverkehrBeleg** nur eine Referenz (z.B. **auftraggeberKontoNummer : KontoNummer**) und nicht das gesamte Objekt (z.B. **auftraggeberKonto : Konto**) angegeben ist.

# Eigenschaften von Beziehungsarten

Beziehungsart \ Kriterium	Assoziation	Aggregation/ Komposition	Generalisierung/ Spezialisierung
<b>Beziehungsname</b>	anzugeben	indirekt vorhanden, da Raute bedeutet: „ist-Teil-von“	indirekt vorhanden, da Dreieck bedeutet: „ist-ein“
<b>Mengenangabe</b>	Ja	Ja	Nein
<b>Gerichtetheit</b>	Ja	Ja	Nein
<b>Rolle</b>	Ja	Ja	Indirekt vorhanden: „Ober-/Unterklasse“ bzw. „Super-/Subtyp“
<b>Beziehungsattribut</b>	Ja	Ja	Nein

# Klassendiagramm - Beispiel

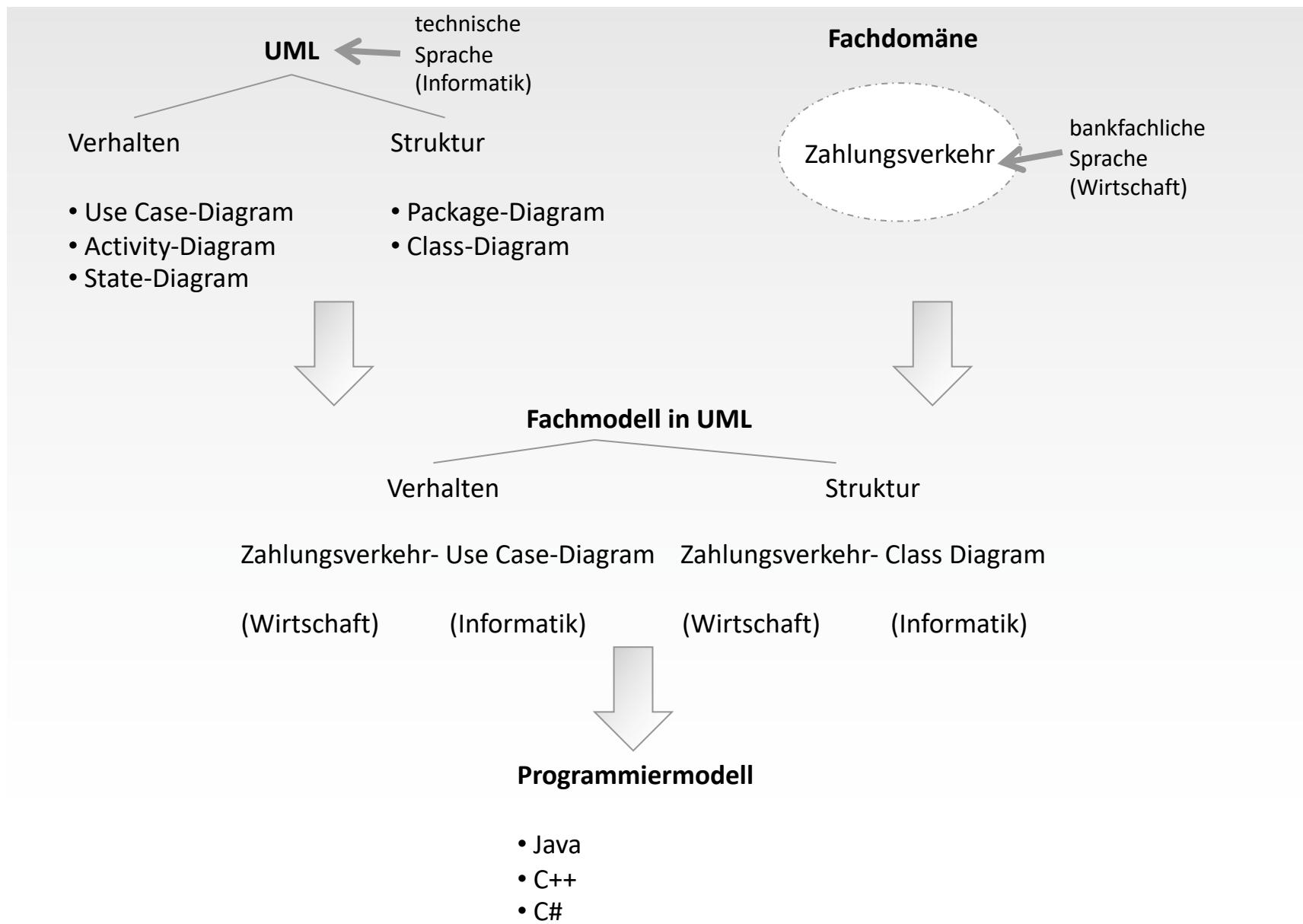
## Fachklassendiagramm

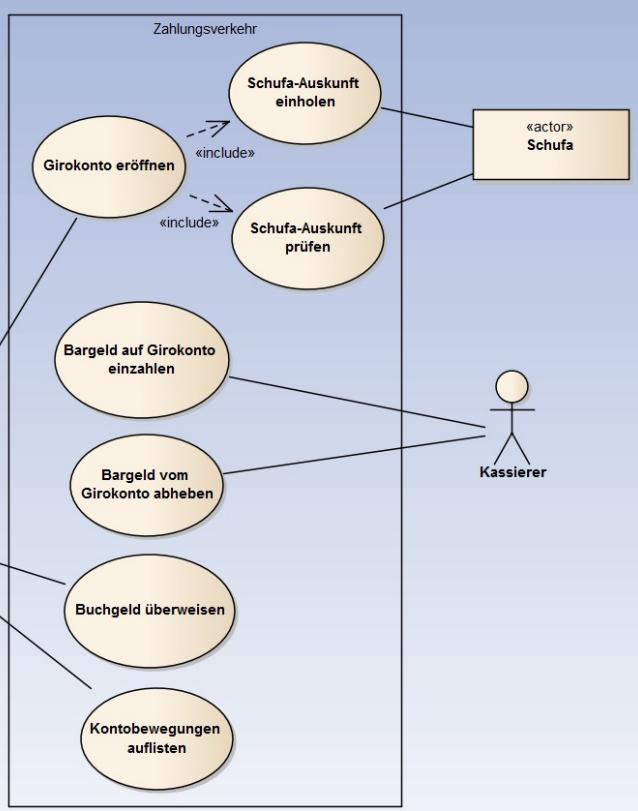


# Von UML zu Java

Voraussetzung: Pflichtenheft liegt vor

1. Erstelle ein Use-Case-Diagramm (korrespondierend zu Funktionen im Lasten-/Pflichtenheft)
2. Erstelle pro Use-Case ein Aktivitätsdiagramm mit Kontroll- und Objektfluss
3. Stelle für Objekte, die in den Aktivitätsdiagrammen vorkommen, potentielle Klassen in einem Fachklassendiagramm (verwende keine Klassen für die grafische Oberfläche)
4. Erstelle Beziehungen zwischen den Klassen des Fachklassendiagramms
5. Reichere das Fachklassendiagramm um Attribute und Methoden an (Statt technischer Datentypen verwende Fachwerte)  
→ Resultat nach den Schritten 3-5 ist ein Fachklassendiagramm, welches die fachliche Struktur der Anwendungsdomäne darstellt
6. Erstelle eine Klasse, die den Namen des Systems im Use-Case-Diagramm hat und füge pro Use-Case eine entsprechende Methode in der Klasse hinzu  
→ Resultat ist eine Klasse, die das fachliche Verhalten der Anwendungsdomäne darstellt
7. Füge die strukturellen Fachklassen dem Paket „Struktur“ zu und gruppiere die Fachklassen unter Verwendung weiterer Pakete
8. Füge die Fachklasse, welche das Verhalten des Systems darstellt, in das Paket „Verhalten“ ein





```

public class Zahlungsverkehr {

  public void eroeffneGiroKonto() {
  ...

}

  public void zahleBargeldEin(ZahlungsverkehrBeleg
      zahlungsverkehrBeleg) {
  ...
}

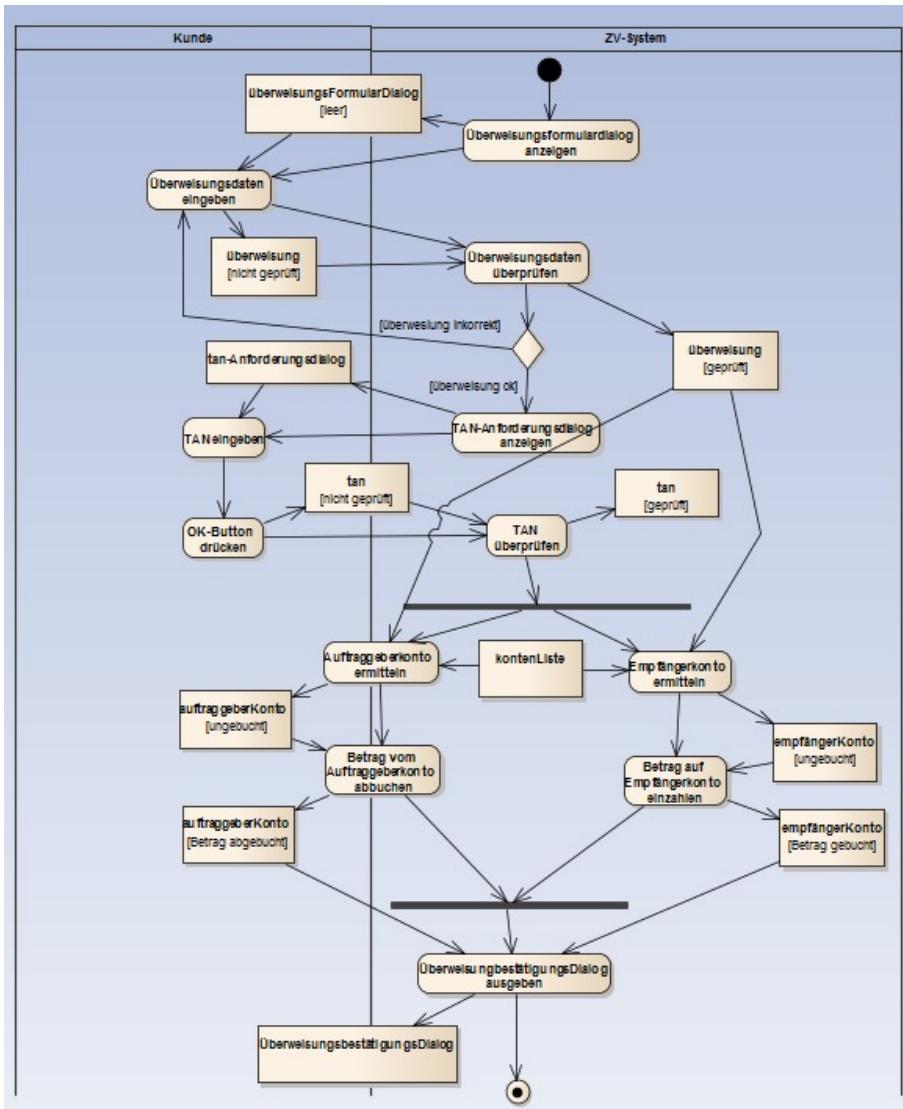
  public void zahleBargeldAus(ZahlungsverkehrBeleg
      zahlungsverkehrBeleg) {
  ...
}

  public void ueberweiseBuchgeld(ZahlungsverkehrBeleg
      zahlungsverkehrBeleg) {
  ...
}

  public void listeKontoBewegungenAuf() {
  ...
}
}

```

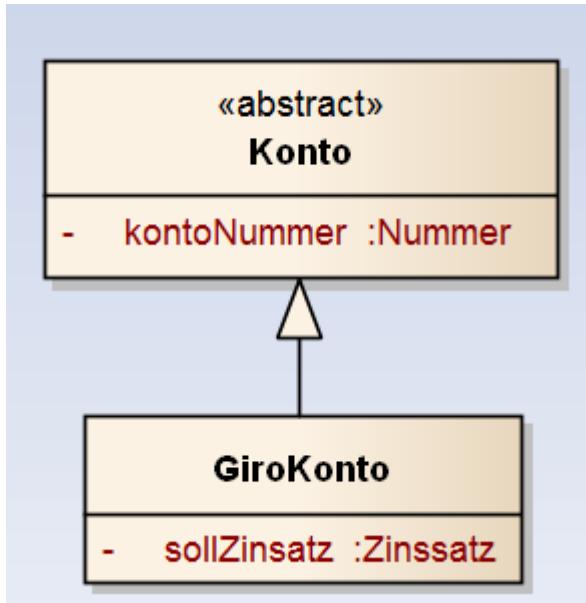
# Von UML zu Java – Activity diagram



```

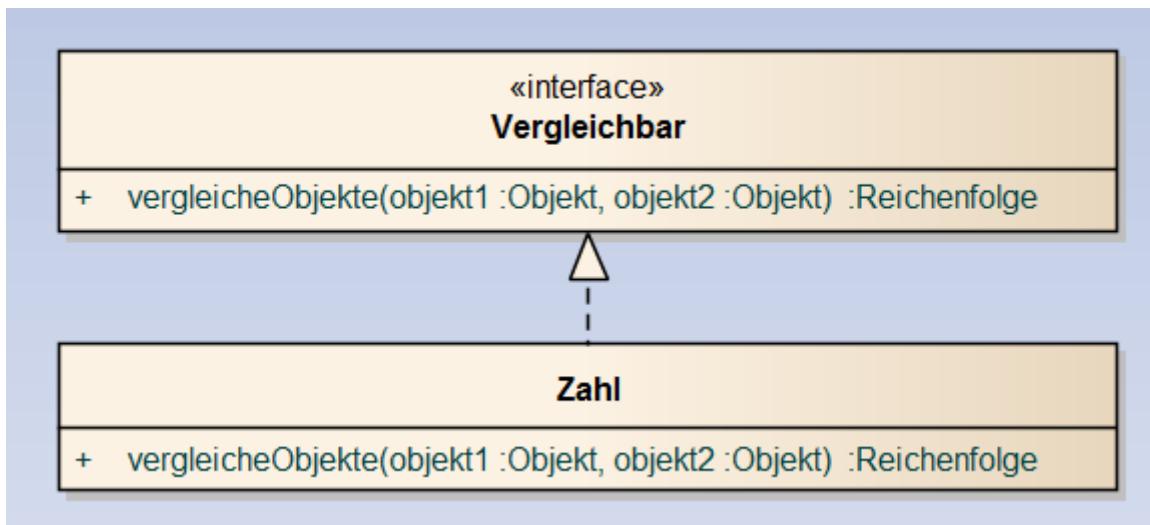
public class Zahlungsverkehr {
    public void ueberweiseBuchgeld(ZahlungsverkehrBeleg
        zahlungsverkehrBeleg,
        Tan tan) {

        // Ueberweisungsdaten prüfen
        ...
        // Tan prüfen
        ...
        // Auftraggeberkonto ermitteln
        ...
        // Empfängerkonto ermitteln
        ...
        // Betrag vom Auftraggeberkonto abbuchen
        ...
        // Betrag vom EmpfängerKonto abbuchen
        ...
    }
}
  
```



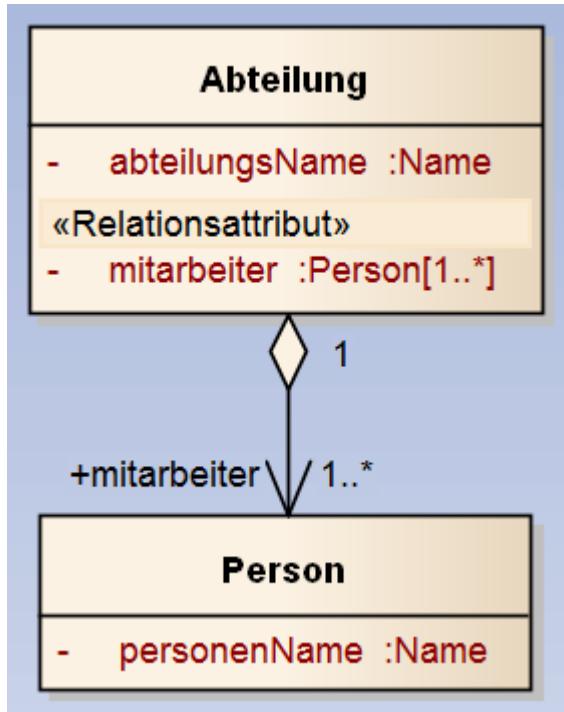
```
public abstract class Konto {  
    private Nummer kontoNummer;  
}
```

```
public class GiroKonto extends Konto {  
    private Zinssatz sollZinssatz;  
}
```



```
public interface Vergleichbar{
    Reihenfolge vergleiche(Object objekt1, Object objekt2);
}

public class Zahl implements Vergleichbar{
    public Reihenfolge vergleiche(Object objekt1, Object objekt2) {
        ...
    }
}
```

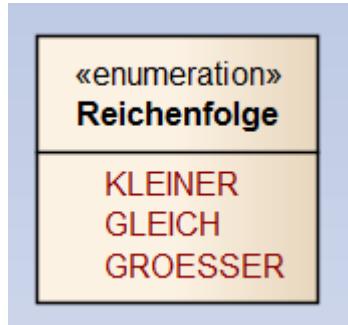


```

public class Abteilung{
    private Name abteilungsName;
    private Set<Person> mitarbeiter;
}
  
```

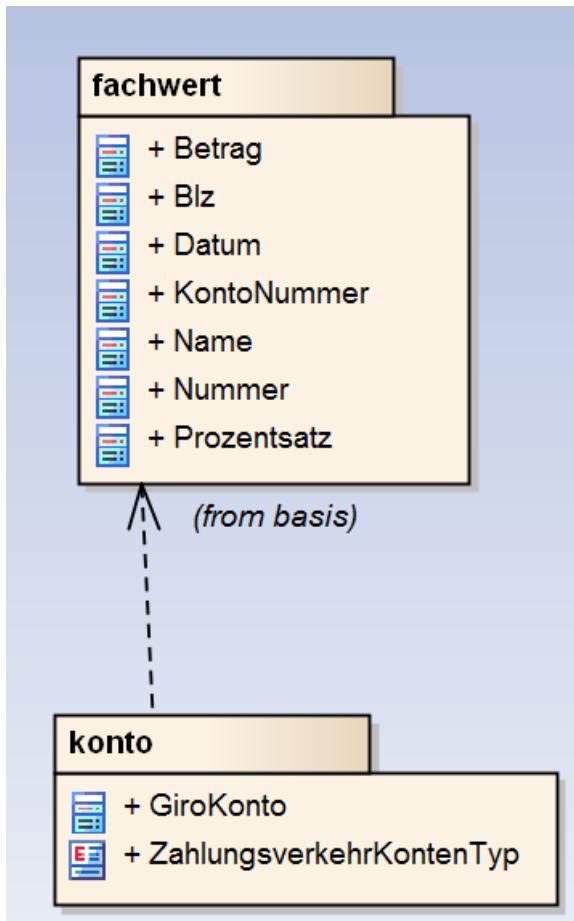
```

public class Person{
    private Name personenName;
}
  
```



```
public enumeration Reihenfolge {  
    KLEINER, GLEICH, GROESSER;  
}
```

# Von UML zu Java – Package diagram – package und dependency

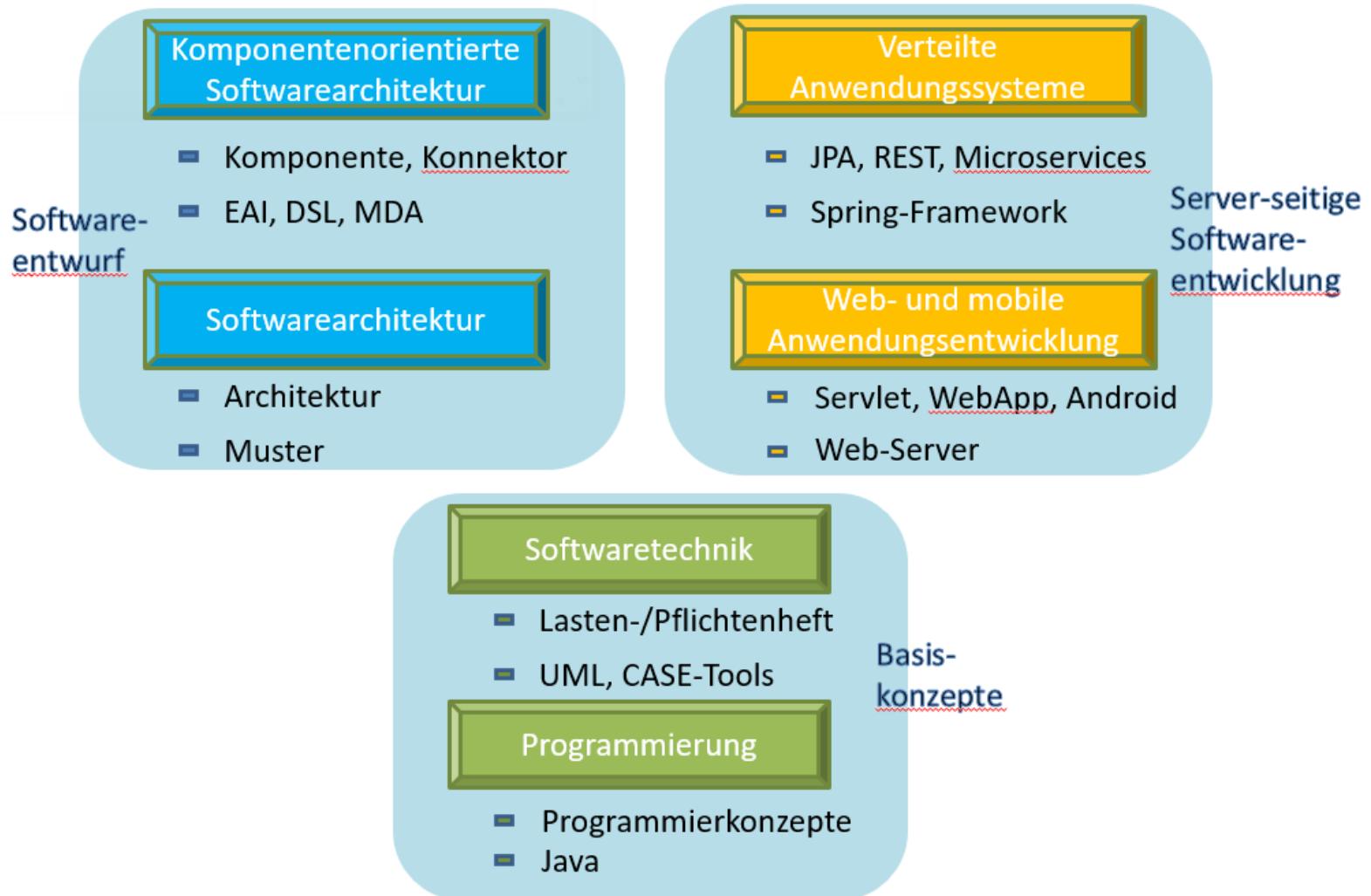


```
package de.leuphana.bank.passiv.zahlungsverkehr.konto;

import de.leuphana.bank.basis.fachwert.Betrag;
import de.leuphana.bank.basis.fachwert.Prozentsatz;
import de.leuphana.bank.basis.konto.Konto;

public class GiroKonto extends Konto {
    private Betrag dispolimit;
    private Betrag ueberziehungsZinsbetrag;
    private Prozentsatz uerberziehungsZinssatz;
    ...
}
```

# **Thematisch weiterführende Vorlesungen**



Im Major „**Wirtschaftsinformatik**“ gibt es verschiedene Vertiefungsmöglichkeiten. Zwei mögliche parallele Stränge sind „**Softwareentwurf**“ und „**Server-seitige Softwareentwicklung**“, die sich auch sinnvoll ergänzen.

Im Strang „**Softwareentwurf**“ macht es inhaltlich Sinn erst „**Softwarearchitektur**“ und dann „**Komponentenorientierte Softwarearchitektur**“ zu hören.

Im Strang „**Server-seitige Softwareentwicklung**“ auch sinnvollerweise erst „**Web- und mobile Anwendungsentwicklung**“ und dann „**Verteilte Anwendungssysteme**“.

Sie können die Vorlesungen im 4. Semester („**Softwarearchitektur**“ und „**Web- und mobile Anwendungsentwicklung**“) bzw. im 5. Semester („**Komponentenorientierte Softwarearchitektur**“ und „**Verteilte Anwendungssysteme**“) hören.

Die Vorlesung „**Softwarearchitektur**“ stellt dar wie Softwaresysteme, die schnell sehr groß werden, durch sogenannte Entwurfsmuster wartbar konstruiert werden können. Entwurfsmuster stellen Strukturierungsvorlagen dar, die sich in der Praxis bewährt haben. Geprägt ist die Softwarearchitektur vor allem durch die Qualitätsanforderungen wie Wartbarkeit, Erweiterbarkeit, Wiederverwendbarkeit u.a. In der Vorlesung soll gezeigt werden wie dies konkret aussieht. Alle theoretischen Erläuterungen werden an einem durchgängigen Beispiel, welches mit UML und Java modelliert bzw. implementiert wird, konkretisiert.

In der Vorlesung „**Komponentenorientierte Softwarearchitektur**“ wird auf der Vorlesung „**Softwarearchitektur**“ aufgebaut. Hier wird gezeigt wie heutzutage durch das Konzept von Komponenten und Konnektoren (ähnlich wie Sie das aus dem Hardware-Bereich kennen) in der Software abgrenzbare und ineinander steckbare Softwarekomponenten und -verbinder (Kabel mit Stecker und Buchse im Hardware-Bereich) hier Software-Konnektoren genannt, konzipiert werden können.

In der Vorlesung „**Web- und mobile Anwendungsentwicklung**“ erfahren Sie wie unter Verwendung von vorhandenen Software-Bibliotheken und dem Einsatz eines Web-Servers ein prototypischer Online-Shop als Web-Anwendung erstellt werden kann. Daneben wird die Entwicklung einer App mit aktuellen Technologien adressiert. Die Vorlesung ist hier, wie der Titel es bereits andeutet, implementierungstechnisch orientiert. Ziel ist es sich mit Technologien und Frameworks, die vorgefertigte und erweiterbare Funktionen anbieten, vertraut zu machen.

Die Vorlesung „**Verteilte Anwendungssysteme**“ stellt standardisierte Technologien in Java im Bereich Connectivity (z.B. Cloud- und Web-Services) vor um verteilte unternehmensweite Softwareanwendungen zu konstruieren. Auch hier werden die einzelnen Technologien an einem konkreten Beispiel (Shop) unter Verwendung des Spring-Frameworks implementiert. Im Gegensatz zu „**Web- und mobile Anwendungsentwicklung**“ handelt es sich um den „nicht sichtbaren Bereich“ (häufig Backend genannt) einer großen Softwareanwendung.

# Literatur

- Balzert, H.; Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering, 3. Aufl., Spektrum Akademischer Verlag, 2009  
<http://dx.doi.org/10.1007/978-3-8274-2247-7> (Zugriff: 18.12.2019)
- Balzert, H.; Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb, 3. Aufl., Spektrum Akademischer Verlag, 2011  
<http://dx.doi.org/10.1007/978-3-8274-2246-0> (Zugriff: 18.12.2019)
- Broy, M., Kuhrmann, M.; Einführung in die Softwaretechnik, Vieweg, 2021  
<https://doi.org/10.1007/978-3-662-50263-1> (Zugriff: 20.03.2022)
- Bruegge, B., Dutoit, A.; Object Oriented Software Engineering Using UML, Patterns, and Java; 3. ed., Prentice Hall, 2010
- Ebert, C.; Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten, 2019
- Ferstl, O., Sinz, E.: Grundlagen der Wirtschaftsinformatik, Oldenbourg, 2013
- Gadatsch, A.; Grundkurs Geschäftsprozess-Management: Analyse, Modellierung und Controlling von Prozessen, Vieweg, 9. Auflage, 2020  
<https://doi.org/10.1007/978-3-658-27812-0> (Zugriff: 12.10.2020)
- Haberfellner, R. et.al.; Systems Engineering: Grundlagen und Anwendung; Orell Füssli, 2012
- Kleuker, S.; Grundkurs Software-Engineering mit UML, 4. Aufl.; SpringerVieweg, 2018  
<http://dx.doi.org/10.1007/978-3-658-19969-2> (Zugriff: 18.12.2019)
- Krallmann, H. et al.; Systemanalyse im Unternehmen: prozessorientierte Methoden der Wirtschaftsinformatik, 6. Aufl., de Gryter, 2013  
<http://www.degruyter.com/doi/book/10.1524/9783486729825> (Zugriff: 18.12.2019)

- Larman, C.; Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd ed., Prentice Hall, 2004
- Ludewig, J., Lichter, H.; Software Engineering, 4. Aufl., dpunkt, 2023
- Metzner, A.; Software Engineering – Kompakt, 2020
- Oesterreich, B., Analyse und Design mit der UML 2.5: Objektorientierte Softwareentwicklung; 11. Aufl.; Oldenbourg, 2013
- Rau, K.-H., Objektorientierte Systementwicklung: Vom Geschäftsprozess zum Java-Programm; Vieweg, 2007
- Ropohl, G., Allgemeine Systemtheorie, Edition Sigma, 2012
- Rupp, C., Requirements-Engineering und –Management, 6. Aufl., Hanser, 2014  
<https://www.hanser-elibrary.com/doi/book/10.3139/9783446443136> (Zugriff: 18.12.2019)
- Seidl, M., et. al.; UML@Classroom: Eine Einführung in die objektorientierte Modellierung; dpunkt, 2012
- Stachowiak, H.; Allgemeine Modelltheorie, Springer, 1973
- Steinpichler, D., Projektabwicklung mit UML und Enterprise Architect, 9.Aufl., SparxSystems, 2012
- Steimann, F.; On the representation of roles in object-oriented and conceptual modelling; Data & Knowledge Engineering, 35 (2000), p.83-106
- Sommerville, I.; Software Engineering, 10th ed.; Addison-Wesley, 2015
- Teubner, A.; Theoretische Grundlagen des Software Engineering, Wirtschaftsinformatik, 05/2002, S. 690-697
- Thomas, O.; Das Modellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation, Institut für Wirtschaftsinformatik, Heft 184, Mai 2005
- Winters, J.; Applying Use Cases – A practical guide, Addison-Wesley, 1998